Classification of Ultra High Range Resolution Radar Using Decision Boundary Analysis

THESIS
Christopher Lawrence Eisenbies
Captain, USAF

AFIT/GE/ENG/94D-07

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

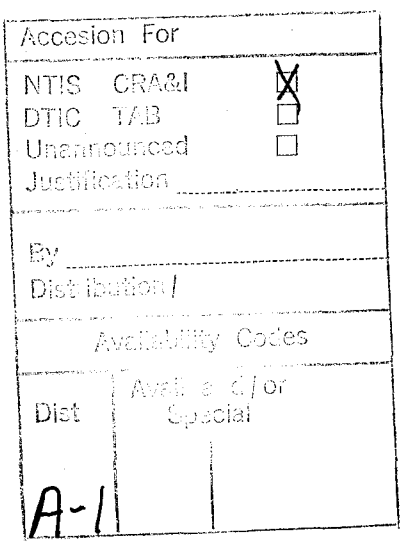
AFIT/GE/ENG/94D-07

Classification of Ultra High Range Resolution Radar Using Decision Boundary Analysis

THESIS
Christopher Lawrence Eisenbies
Captain, USAF

AFIT/GE/ENG/94D-07

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.



AFIT/GE/ENG/94D-07

Classification of Ultra High Range Resolution Radar Using Decision Boundary Analysis

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Christopher Lawrence Eisenbies

Captain, USAF

December, 1994

Acknowledgements

The completion of this thesis would not have been possible without the insights and patience of my thesis advisor, Doctor Steven Rogers. Even though he is a Tarheel at heart and myself a "Dookie," he found a way to reawaken my intellectual excitement and curiosity. I'd like to recognize the significant amount of steady, good advice provided by Captain Tom Burns, who was essential to keeping everything on track and in perspective. Also, the contributions of Captain Dennis Ruck, with his timely and crystal clear help with probability theory; and Major Gregory Warhola, who taught me all the essentials of wavelets. The list continues with the early input of Pete Kosir and Rob DeWall of VEDA and Rick Mitchell from WL/AARA, all of whom helped clarify the AGC and its approach to the UHRR problem. Laura Suzuki deserves special thanks for her comments with respect to multiresolution tidbits and the use of the computer code she wrote. Thanks also goes to Lem Myers for the use of his code as well.

Christopher Lawrence Eisenbies

Table of Contents

List of Figures

List of Tables

Abstract

This thesis examines the discrimination of targets with Ultra High Range Resolution (UHRR) radar data. Using these measured signals from frontal aspect angles of four aircraft classes, the baseline performance of the Adaptive Gaussian Classifier (AGC) is tested with respect to aligning exemplars to templates. Alignment plays a crucial role in the AGC's classification performance which can degrade by 11% for a target class. The AGC is compared to non-parametric classifiers, but no statistically significant degradation of performance is found. Data separability is analyzed by bounding the Bayes error. The data is well separated in a statistical sense. A feature selection algorithm, based on analysis of the decision boundary, is applied to find a reduced feature set, which are linear combinations of the original features. These features are optimized with respect to classification error rather than reconstruction error. This technique is extended to deduce the relevant features in the *original* feature space. Fewer than 5% of the features in the original feature space may be used to attain an improved classification rate. This new method is a true reduction of features and shows improvement up to 15%. Discrimination of UHRR radar signatures using a multiresolution analysis is proposed. The decision boundary analysis chooses relevant wavelet scales with respect to classification. Some improved performance against an entropy based measure is observed for limited feature sets. The technique developed here successfully chooses the scale that causes classification performance to peak within 5% of the performance in the full-dimensional or reduced-dimensional UHRR radar signature space.

Classification of Ultra High Range Resolution Radar Using Decision Boundary Analysis

I. Introduction

1.1 Background

Non-Cooperative Target Identification (NCTI) is a top-priority area of research for the Air Force. NCTI is an Automatic Target Recognition (ATR) approach that involves the ability to identify targets without active transmitters or transponders on the target. ATR enables a searcher to distinguish between friendly (blue) and other forces (red/gray). In the case of identifying blue systems, active devices, such as Identification, Friend or Foe (IFF) transmitters are used, but they cost money to develop and add a point of vulnerability on the battlefield. A reliable NCTI capability holds the promise of less risk of intercept of tell-tale electromagnetic (EM) signals on the battlefield. Also, the need for the design and procurement of such IFF devices is eliminated. One common, non-cooperative source of information about a target lies embedded in its radar signature. Reflected EM waves are a readily accessible source of information unique to the geometry and composition of a target. The ongoing challenge is to find ways to extract and use this information effectively.

Ultra High Range Resolution (UHRR radar) radar is a type of radar that can be used for NCTI (34, 44, 1). As in all radar processing techniques, total energy returned from a target is distributed into "range bins." Range bins contain the energy reflected from target structures at incremental distances from the radar source. Using UHRR radar signatures for NCTI is based on the premise that signatures contain unique information about a target class. This premise is intuitively attractive for ATR because one may use this unique information for recognition if one can discover where the discriminantly relevant information lies.

Statistical pattern recognition uses data extracted from targets, such as UHRR radar signatures, to classify them. Tools include parametric (Gaussian classifiers) and non-parametric (k-nearest neighbor and neural network) systems. A Gaussian classifier is currently being pursued in the UHRR radar problem at WL/AARA. This approach makes important assumptions about the statistics of the data. Fukunaga and Martin applied non-parametric density estimations to computer generated UHRR radar signatures, making predictions about the underlying information content in a signature (20, 33). Earlier Air Force Institute of Technology students, Dewitt and Kouba, used Hidden Markov Models and recurrent neural networks to classify targets (26, 15). Hughes aircraft developed an Adaptive Gaussian Classifier (AGC) which is currently being exercised at Wright Laboratories. In the general case, the AGC is adaptive in the sense that mean and variance computations are adjusted to compensate for amplitude changes within the incoming signals (34:12).

The critical problem in pattern recognition is how to extract data (or features) from targets efficiently. "Efficiently" has a dual meaning. First, it means transforming or projecting the raw data into a convenient set of parameters (*i.e.* features) representing that object. The parameters must be energy normalized and properly aligned for comparison. "Efficiently" also means using only those features that are relevant to distinguishing classes of targets for the problem at hand. Every feature included in an analysis has a computational cost associated with it. Also, it is possible to include too many features, saturating the capability of the classifier for the amount of training data available (23:204). *A priori* knowledge of the problem may be used to help choose effective, discriminating features.

1.2 Problem Statement

This thesis investigates raw UHRR radar signatures as feature vectors in the classification of aircraft. The sensitivity of classification error to proper alignment, radar "flashes," and choice of classifier is studied. An important, related problem is identifying those features which contain information relevant to successful classification. A technique for doing so is ap-

plied and modified to choose an efficient set of features in the raw data and in a multiresolution wavelet decomposition of the data.

1.3 Scope

This thesis analyzes the ability of the AGC to properly align and classify efficient features in the context of UHRR radar data. The situation analyzed is a four class problem with approximately 1000 signatures of measured data available for each class. The current AGC will be baselined by controlling which signatures the AGC will train on and classify. This test will show some signatures contain radar "flashes" that cause the AGC to improperly align signatures with the correct class template. The utility of a Gaussian classifier in this pattern recognition problem is tested by comparing its performance to non-parametric classifiers with the same data sets. The information content of the radar returns is explored by bounding the Bayes error. Finally, discriminant analysis is applied to the problem using the decision boundary generated by the quadratic Gaussian classifier. The goal is to determine the discriminantly relevant portions of the signatures. In a two class situation, it is shown that specific frequency bands, generated with a wavelet decomposition contain most of the pertinent information for classification.

1.4 Approach

This thesis begins by baselining the Hughes' AGC as developed for the NCTI program at Wright Laboratories. Classification rates are estimated by training and testing using the holdout method and repeating over many independent trials (19:220-221). The impact of feature vectors which are not properly aligned by the AGC is analyzed by withholding signatures that are not properly aligned from the classification process and seeing whether classification improves. The effects of alignment are analyzed by hand aligning the problem signatures and then using the AGC. The classification rates for each case are compared to see which effects have a significant impact on the classification rate. A k-Nearest Neighbors (k-NN) classifier and a neural network are used to test whether the statistical nature of the UHRR radar data

may be better represented by a non-parametric classifier. Class separability is measured by performing a Bayes Error Bounding test as described by Fukunaga and implemented by Martin (20, 33).

After assessing these questions, feature discrimination is explored. First, the idea that some features are more relevant to classification than others is examined. Then, the analysis investigates the premise that information relevant to a target can be found in distinct bands of the frequency domain. The tool used to explore this idea is the wavelet, because wavelet decompositions systematically break out the frequency domain into orthogonal bases. The basic idea originates from Coifman (10:714) and measures the information content of the bands with an entropy measure. Chang applies the l_1 norm as an entropy measure and uses the results for class discrimination with respect to two dimensional textures. This thesis finds discriminantly relevant bands with a modified version of a discrimination technique described by Lee and Landgrebe (29).

1.5 Objectives

There are five objectives of this thesis:

1. Describe and document the implementation of the Hughes AGC as provided for this research. (Chapter II, Section 2.5)
2. Examine the effect of proper alignment during training and testing on the AGC's performance. Examine the effect of signatures with noise flashes (corrupt signatures) even when properly aligned to class templates. (Chapter III, Section 3.3)
3. Determine whether a non-parametric classifier may be better suited to the UHRR radar problem. (Chapter III, Section 3.4)
4. Quantify whether the UHRR radar data is separable by estimating the Bayes error. (Chapter III, Section 3.5)

5. Extract discriminantly relevant features with the raw data and with a multiresolution analysis to improve classification or reduce the number of features required for similar performance. (Chapter III, Section 3.7 and Chapter V, Section 5.2)

1.6 Organization

Chapter II begins with a description of the statistical and mathematical theories which support the methodology and results of this thesis. Also included is a description of how the AGC works. Chapter III describes the evaluation of the AGC with respect to the alignment, registration, and corrupted signature issues. Bayes error analysis is used to provide a measure of classification capability and class separability. Feature analysis is used to quantify where in the signatures the discriminantly relevant information lies. Chapter IV presents results for the statistical and feature analyses described in Chapter III. Chapter V explores the use of wavelet multiresolution analysis to find those frequency bands where discriminantly relevant information lies.

II. Theory

2.1 Introduction

The underlying mathematics behind extracting features and discriminating in an N dimensional feature space are well understood within the context of statistical pattern recognition (19). A key factor is to find the balance between computation time and classification accuracy. Another key element is having enough sample data to establish a reasonable representation of class boundaries for the data being analyzed.

Linear algebra is the tool used to frame the problem. Representations of signals (features) may be imagined in an N dimensional Euclidean space, R^N, where N is the number of features. By extracting features from a signal, a feature vector is projected into R^N and is denoted as a column vector, $\underline{x} = [x_1 x_2 \ldots x_N]^T$. Samples of a class of target would be expected to share common features (and, thus, identical feature vectors), but noise causes actual measurements to vary within R^N. For the UHRR radar problem, noise comes from uncertainties in relative positions of target and radar, atmospheric effects, and equipment variations. Probability density functions (pdf's) are estimated and used to find where in R^N a given class tends to cluster. In some cases, these pdf's may also be used to generate class boundaries and indicate class separability. When an unknown signature (exemplar) is presented to the classifier, the classifier "guesses" the proper class assignment based on the estimated parameters of the underlying statistical distributions.

Theoretically, the limit to this success rate is known as the Bayes error rate (41:35). The Bayes decision rule tells one where to draw the decision boundary lines in the feature space. Boundaries between classes are formed and used to classify unknown observations. In the case of parametric classifiers, the boundary may be estimated analytically. One recently developed algorithm uses information derived from the estimated decision boundary itself to determine a new set of relevant features. These topics are discussed in the first half of this chapter. Also included is a brief introduction to wavelets, which will be used as a tool in demonstrating frequency analysis in a two class case.

With the Adaptive Gaussian Classifier (AGC) algorithm the assumption is, naturally, the distributions of the data are Gaussian. For the purposes of this thesis, AGC refers to the entire Hughes algorithm, including preprocessing and classification. UHRR radar signatures are projected into a 192-dimensional feature space, R^{192}, where the complex return in each range bin represents a feature. This strategy is validated by Fukunaga, who has shown the best you can ever do, classification wise, is with the raw data (19). Any processing or filtering ultimately removes some information from the data. Several preprocessing steps are taken to put the data into usable form. The AGC takes the data, aligns it to class template vectors, and classifies signatures with a Gaussian classifier. The last portion of this chapter details the steps in that process.

2.2 *Bayes Decision Rule*

2.2.1 Fundamentals. The foundation of statistical pattern recognition rests on Bayes rule which is expressed mathematically as (19, 41)

$$p(\omega_i|x) = \frac{p(x|\omega_i) \cdot P(\omega_i)}{p(x)}. \tag{1}$$

Bayes rule combines information about the *a priori* probability of a class, $P(\omega_i)$, with the total probability, $p(x)$, of a measurement value, x, and the conditional probability that x belongs to ω_i: $p(x|\omega_i)$. The *a posteriori* pdf, $p(\omega_i|x)$ expresses the probability that the unknown sample belongs to class ω_i, given a measurement x.

The pdf for a continuous random variable x, when integrated between two limits, gives the probability that x will takes on a value between those limits. The pdf for a discrete random variable yields a similar value when the pdf is summed across the indices included by the limits. The right hand side of Equation 1 includes three terms. $P(\omega_i)$ is the *a priori* probability that class ω_i occurred. That is, for an infinite number of trials, it is the expected frequency that ω_i has appeared. The term $p(x|\omega_i)$ is the conditional pdf yielding the probability that x is observed, given that the i^{th} class occurred (ω_i, $i \in \{1, 2, ...K\}$ and K is the number

of classes). $p(x)$ is the overall pdf for a measurement x: in other words, a measure of the expected value of x. Note that

$$p(x) = \sum_{i=1}^{L} p(x|\omega_i) \cdot P(\omega_i), \tag{2}$$

which says that the overall pdf for x equals the sum of each class's conditional pdf multiplied by the *a priori* probability for that class. One term of this summation (the i^{th} class's contribution to the sum) is used in Equation 1. The ratio of that portion of the summation to the overall pdf of x gives the *a posteriori* probability of class ω_i.

Bayes decision rule assigns an unknown measurement, x, to the class with the highest *a posteriori* value. This criterion insures that the probability of assigning an exemplar to the wrong class is minimized and is easily extended to the multivariate case where x need not be a one-dimensional vector, but could be any N-dimensional vector, $\underline{x} \in R^N$, containing the features extracted by some measurement.

For a two-class problem, the probability of error is equal to the chance that the classifier guesses the wrong class, given a value of $\underline{x}$. Mathematically, one integrates the $\underline{x}$ dependence out of the conditional probability:

$$P(\text{error}) = \int_{-\infty}^{\infty} P(\text{error}|\underline{x})p(\underline{x})d\underline{x}, \tag{3}$$

to get the expected probability of error for a data set. To account for errors over both classes, substitute expressions representing the probabilities of making an error for each class into Equation 3 (19),

$$P(\text{error}) = \int_{S_2} p(\underline{x}|\omega_1)P(\omega_1)d\underline{x} + \int_{S_1} p(\underline{x}|\omega_2)P(\omega_2)d\underline{x}. \tag{4}$$

where the S's represent the respective decision regions for each class. In words, it adds up the probability of an exemplar occurring in the wrong decision region. Note that the dependence on $p(\underline{x})$ has divided out. As stated in Fukunaga (19), this is the minimum Bayes error rate

for any classifier. Equation 4 expresses mathematically the rule of calculating the area under the tails of the pdf's for finding the expected probability of error. See Schalkoff (41:35) for further details.

2.2.2 Functional Approximations of pdf's. The *actual* underlying pdf of a random process (such as a UHRR radar signature) is rarely known. The conditional pdf, $p(\underline{x}|\omega_i)$, for a class, ω_i, is estimated or parameterized, based on samples from that class. There are two approaches for constructing the conditional pdf: parametric and non-parametric.

2.2.2.1 Parametric. Parametric approximations assume a functional form for the pdf and calculate the function's parameters based on a set of sample points, called the training set. The most significant example is the Gaussian pdf. The Gaussian is important because many natural processes tend to have Gaussian distributions. The *central limit theorem* states that the statistics of a system, no matter what the underlying random variables contributing to the process, becomes Gaussian as the number of underlying variables goes to infinity (19:17). The *central limit theorem* presumes that the process is a linear combination of the underlying random variables. Note the Gaussian case assumes a unimodal pdf, which may or may not be the true "shape" of the actual pdf. This assumption will cause problems if it does not reflect the true distributions of the data.

The parameters that completely define a one-dimensional Gaussian pdf are the mean (μ) and variance (σ^2) of the associated random variable. Each is estimated using the training set and the training procedure forms an *estimate* of the pdf of the actual random process. These estimates should be "unbiased" so that the random process which generates the data is properly characterized. An estimate is unbiased when $\hat{\mu}$ and $\hat{\sigma}^2$ converge in a mean square sense to the expected values of μ and σ^2 when $J \rightarrow \infty$, where J is the number of samples used in training. In other words, the estimates should converge to the actual values as one uses more and more training samples.

To form these estimates, suppose the training data consists of measurements x_j or $\underline{x}_j$, where j is the j^{th} sample from a set of J samples. For one dimension, unbiased estimates of

μ and σ^2 are (19)

$$\hat{\mu} = \frac{1}{J}\sum_{j=1}^{J} x_j \tag{5}$$

$$\hat{\sigma}^2 = \frac{1}{J-1}\sum_{j=1}^{J}(x_j - \hat{\mu})^2. \tag{6}$$

Recalling that i is the i^{th} class, the estimated Gaussian pdf is then

$$p(x|\omega_i) = \frac{1}{\sqrt{2\pi}\hat{\sigma}_i^2}\exp\left[-\frac{1}{2}\left(\frac{x-\hat{\mu}_i}{\hat{\sigma}_i^2}\right)^2\right]. \tag{7}$$

Similarly, an N-dimensional feature space (the multivariate case) is handled with vectors in the following version of Equation 7 :

$$p(\underline{x}|\omega_i) = (2\pi)^{-\frac{N}{2}}|\hat{\Sigma}_i|^{-\frac{1}{2}}\exp\left[-\frac{1}{2}\left(\underline{x}-\underline{\hat{\mu}}_i\right)^T\hat{\Sigma}_i^{-1}\left(\underline{x}-\underline{\hat{\mu}}_i\right)\right], \tag{8}$$

where $|\hat{\Sigma}_i|$ represents the determinant of the estimated covariance matrix of ω_i and

$$\underline{\hat{\mu}}_i = \frac{1}{J}\sum_{j=1}^{J}\underline{x}_j \tag{9}$$

$$\hat{\Sigma}_i = \frac{1}{J-1}\sum_{j=1}^{J}(\underline{x}_j - \underline{\hat{\mu}}_i)\cdot(\underline{x}_j - \underline{\hat{\mu}}_i)^T. \tag{10}$$

For future reference, note that this equation is an averaged sum of outer products. The covariance matrix, $\hat{\Sigma}_i$, gives the statistical interrelationship between the feature elements (37:427-455). Diagonal elements are variance estimates. If all off-diagonal elements are zero, the features are statistically independent.

The exponent term on the end of Equation 8 is an important one and is known as the Mahalanobis distance between the measured data $\underline{x}$ and the estimated mean of class ω_i. Unlike a Euclidean distance, the Mahalanobis distance normalizes with respect to variance

that feature axis in R^N. Thus, distances between features which have widely varying values are normalized to distances between features with more "concentrated" pdf's.

2.2.2.2 Non-parametric. Non-parametric approximations assume no form for the pdf function as a whole. Instead, the pdf is estimated directly from the data. One type of non-parametric classifier is the *k*-NN, where the idea is that the pdf will be larger where more samples tend to appear and smaller where fewer samples appear. Although specific definitions may be argued, another type of non-parametric technique is the neural network. Martin gives an excellent description of estimating pdf's with these techniques (32, 33).

The *k*-nearest neighboring technique is a common non-parametric density estimate. As developed by Fukunaga (19:268), a random hyper-volume, $\mathbf{V}$, is created around a given training vector, $\underline{x}$. This volume expands until the k^{th} nearest neighbor matching $\underline{x}$'s class, ω_i, is found. Thus, $\mathbf{V}$ is a random function of $\underline{x}$ and is smaller in regions where samples of ω_x are most dense. This is an inverse relationship which may be expressed as

$$p(\underline{x}|\omega_i) = \frac{k-1}{N\mathbf{V}}. \tag{11}$$

where N is the total number of samples and k is the number of nearest neighbors used when forming $\mathbf{V}$. Any distance metric may be used to judge which neighbors are closest to $\underline{x}$.

Artificial neural networks may also be used to estimate the *a posteriori* probabilities of the classes in a pattern recognition problem. The term "neural network" originates from the behavior of neurons in the brain, but the association is oblique, at best. A brief overview of neural networks is included in this thesis. This overview is in the same form as found in Martin (32), and further details may be found in Lippmann (30) and Rogers (39).

A neural network consists of a matrix of nodes which are interconnected by weighted input and output paths. The weights are adjusted during training, when they are modified to force the network to yield a desired output for a given set of training vectors. A basic neural network architecture, known as a multi-layer perceptron, is shown in Figure 1. Using the

standard terminology, this particular network has one hidden layer with five hidden nodes. Only four of the weights (represented by the interconnecting lines) are labelled, but each connection has a weight associated with it. There are three input nodes and two output nodes. Also shown are bias nodes, represented as the 1's enclosed in circles. These nodes are an integral part of the system and have weights associated with their interconnections like all the other nodes.

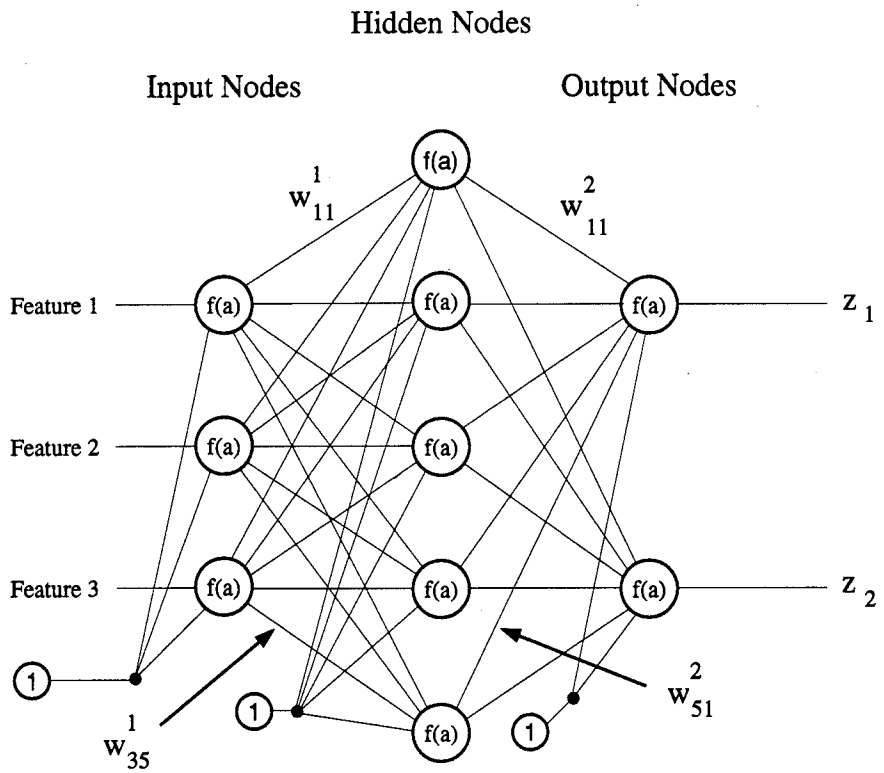


Figure 1. Example of a Multi-Layer Perceptron

To use a neural network as a classifier, one may require that z_1 be "high" relative to z_2 when a member of class ω_1 is presented at the input nodes. One may have 192 input nodes and four output nodes for the four-class UHRR radar problem. The number of hidden nodes is fairly flexible, but depends on the complexity of the decision regions required to separate the data. Usually, one must vary the number of hidden nodes as a parameter and simply see how results are affected. This problem is similar to the problem of choosing the proper number

of nearest neighbors to use in the k-NN algorithm. See Martin (33) and Fukunaga (20) for discussions of this issue.

During learning, training vectors are presented to the neural network inputs. Each input is weighted by w_{ij}^k and then presented as addends to the hidden nodes. k is the k^{th} layer, i is the i^{th} input to that node, and j is the j^{th} node at the k^{th} layer. The weights are initially randomized. Each hidden node implements a non-linear function known as a sigmoid. Other functions may be used, but sigmoids have attractive characteristics which are mentioned in the literature (39). Mathematically, for each node,

$$f(a) = \frac{1}{1+e^{-a}}, \tag{12}$$

where a is a weighted sum of outputs from the previous layer. Weights are usually updated by a process known as "backpropagation." This gradient descent technique seeks to minimize an error measure for a given training exemplar and set of weights. A common error measure is the square error between desired and actual outputs for a given output node. As updated values of the weights are generated, the neural network is conditioned to provide the appropriate outputs for a given class at the input nodes.

As shown by Ruck (40), a neural network trained in this way approximates the underlying *a posteriori* probability function. Rogers shows how the internal representations of the weights may be combined and interpreted as decision boundaries for classification (39:58). The bias nodes allow the decision boundaries to not necessarily pass through the origin. Steppe provides a rigorous method for determining salient features from classification by a neural network (42, 43).

2.2.3 Likelihood Ratios. Fukunaga discusses the use of pdf estimates to make optimal decisions in a given pattern recognition problem (19:51). The following development parallels his. Decisions in statistical pattern recognition depend upon the pdf estimate used, the training data available, and the distance metric used to measure the similarities between test samples and templates. These dependencies require the researcher to be careful in setting

up experiments. The three dependencies mentioned here parallel the questions to be explored in Chapter 3 of this thesis.

One chooses the class of maximum likelihood in a two class problem by assigning a test vector to a class. Using Equation 1,

$$\frac{p(\underline{x}|\omega_1) \cdot P(\omega_1)}{p(\underline{x})} \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \frac{p(\underline{x}|\omega_2) \cdot P(\omega_2)}{p(\underline{x})}. \tag{13}$$

The above equation says "decide class 1 if the left hand side is greater than the right hand side of the equation and decide class 2 if the left hand side is less than the right hand side of the equation." Because it is a pdf, $p(x)$ is by definition always positive and is common to both sides of the equation, so it divides out, leaving

$$p(\underline{x}|\omega_1) \cdot P(\omega_1) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} p(\underline{x}|\omega_2) \cdot P(\omega_2) \tag{14}$$

$$\frac{p(\underline{x}|\omega_1)}{p(\underline{x}|\omega_2)} \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \frac{P(\omega_2)}{P(\omega_1)}, \tag{15}$$

with

$$\mathcal{L}(\underline{x}) = \frac{p(\underline{x}|\omega_1)}{p(\underline{x}|\omega_2)}. \tag{16}$$

$\mathcal{L}$ is known as the "likelihood ratio". Assuming Gaussian distributions, versions of Equation 8 (corresponding to the two classes under test) are inserted into Equation 15 and a natural logarithm is taken to remove the exponential and yield

$$-\Big(.5(\underline{x}-\hat{\underline{\mu}}_1)^T\hat{\Sigma}_1^{-1}(\underline{x}-\hat{\underline{\mu}}_1) - .5(\underline{x}-\hat{\underline{\mu}}_2)^T\hat{\Sigma}_2^{-1}(\underline{x}-\hat{\underline{\mu}}_2) \tag{17}$$

$$- .5\ln\frac{|\hat{\Sigma}_2|}{|\hat{\Sigma}_1|}\Big) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \ln\frac{P(\omega_2)}{P(\omega_1)} \tag{18}$$

$$\Longrightarrow \tag{19}$$

$$-\ln \mathcal{L}(\underline{x}) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \ln \frac{P(\omega_2)}{P(\omega_1)}. \tag{20}$$

This equation represents a quadratic surface in R^N which separates the two classes. Using this equation as a classifier presumes unimodal Gaussian distributions, but is analytically tractable.

The decision rule or discriminant function, $h(\underline{x})$ is defined to be (29):

$$h(\underline{x}) = -\ln \mathcal{L}(\underline{x}). \tag{21}$$

The minus sign introduced here cancels with the minus signs in the left hand portion of Equation 18, and the inequalities change directions by exchanging numerator and denominator on the right hand side of the equation. These steps proceed as follows:

$$h(\underline{x}) \underset{\omega_2}{\overset{\omega_1}{\lessgtr}} \ln \frac{P(\omega_2)}{P(\omega_1)} \tag{22}$$

$$h(\underline{x}) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \ln \frac{P(\omega_1)}{P(\omega_2)} \tag{23}$$

$$h(\underline{x}) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} t. \tag{24}$$

The parameter t is known as the decision threshold. If *a prioris* are assumed equal, t becomes zero. The discriminant function, $h(\underline{x})$ has a nice intuitive appeal because it implies that $\underline{x}$ "probably belongs" to the nearest class template using the Mahalanobis distance metric, plus an additional term. This additional term contains the ratio of the determinants of the covariances and biases the decision in favor of the class with the tighter distribution. Thus, the determinant term will be referred to as the "bias" for the remainder of the thesis. Strang shows how the determinant can be interpreted as a volume in N-dimensional space (45). If a class is spread out or tends to "occupy" a significant volume in R^N, that class is penalized. For a K-class problem, one takes the minimum of the K interclass distances (scaled by the bias) in making a decision. Note that in Fukunaga (19), the decision rule has a typographic error throughout the book because the greater than/less than symbol is reversed from what it should read (for example, see Page 125, Equation (4.1)).

$h(\underline{x})$ may also be computed in the case of k-NN's. The following equations are taken from Martin's thesis (32:14-15) and are implemented later in his code. Use Equation 11 and insert the equation for V:

$$V = V_d|\Sigma|^{1/2}m_d(\underline{x}). \tag{25}$$

In this equation, m_d is the square root of the Mahalanobis distance between the exemplar and the class mean being tested against. V_d is a hyperellipsoid of N dimensions which is given by

$$V_d = \frac{\pi^{n/2}}{(n/2)!}, \; n \text{ even}, \tag{26}$$

$$V_d = \frac{2^n \pi^{(n-1)/2} \left(\frac{n-1}{2}\right)!}{n!}, \; n \text{ odd}. \tag{27}$$

These equations yield

$$p(\underline{x}|\omega_i) = \frac{k-1}{NV_d|\Sigma_i|^{1/2}m_d(\underline{x})}. \tag{28}$$

and may be inserted into Equation 15.

Bayes decision rule, developed above, is the core of statistical pattern recognition. As stated by Fukunaga, "the probability of error is the most effective measure of a decision rule's usefulness (19:85)," and Equation 4 is the equation giving that value. Unfortunately, that integral can be analytically intractable in high dimensional problems even with the Gaussian assumption. To overcome this, Fukunaga suggests using a Monte Carlo analyses or bounding the error probabilities (19:85). In this thesis, both techniques are utilized.

In any case, one is estimating the predicted error rate for the classifier. In effect, the estimate itself is a random variable and has some variance associated with it. This uncertainty may be quantified with the use of confidence intervals. When applying a confidence interval to an error rate, one is estimating the variance of a proportion, where the proportion is a measure

of the expectation of a correct classification. The proportion behaves as a Bernoulli random variable with a binary output: correctly classified or not correctly classified (13:347).

Any statistics book will give the basics of this procedure, but Papoulis (37:241-256,270) gives the complete story, if one is careful to follow his cross-references. The technique treats the classification rate produced from one Monte Carlo trial as an estimate of the mean of a Bernoulli process. As more estimates of the mean are taken, one becomes more confident that the actual mean is "close to" this value.

To produce a confidence interval, one requires a test statistic, $z_{a/2}$. The degree of confidence is "$1 - a$ ", in terms of a percentage. As shown in Figure 2, the goal is to find

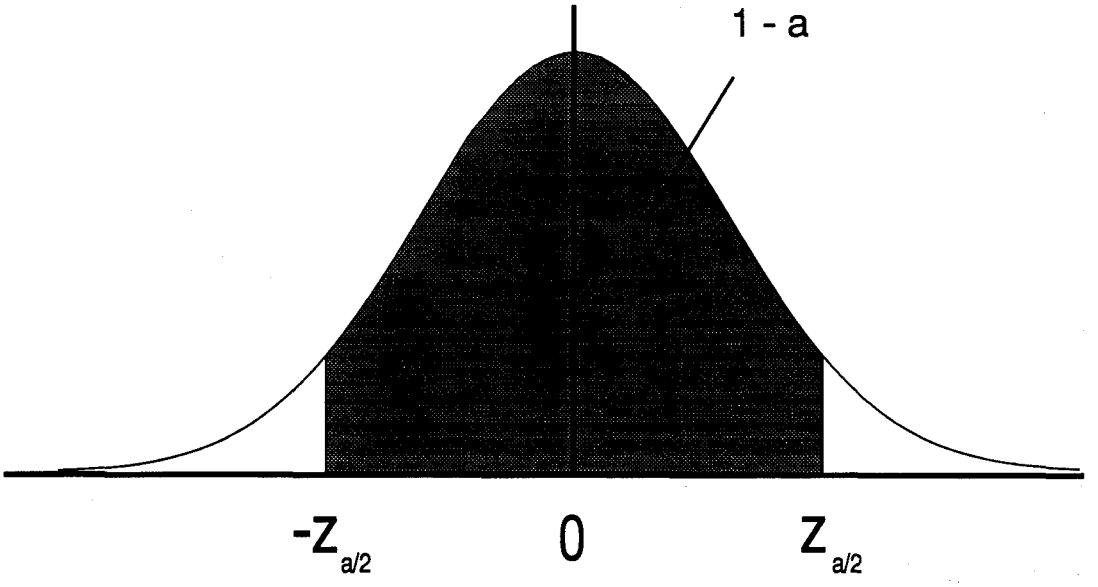


Figure 2. Confidence Interval Schematic

the extent of the shaded region of the curve, which represents the density function of the Bernoulli, or binomial, random variable. In this case, the region shown may represent where 97.5 % of the mean estimates are expected to fall, after normalizing the mean to zero. This implies $a/2$ equals 0.0125. The value of $z_{a/2}$ may be obtained from standard tables and for a

97.5% confidence level, $z_{a/2} = 1.96$ (2:624). The variance is estimated with

$$\hat{\sigma}^2 = \frac{\hat{p}(1-\hat{p})}{n}, \tag{29}$$

where $\hat{p}$ equals the average classification rate for n trials. The following equations are used to find the lower and upper bounds on the estimate:

$$P\left(-z_{a/2} < \frac{\hat{p}-p}{\sqrt{\hat{p}(1-\hat{p})/n}} < z_{a/2}\right) = 1-a$$

$$P\left(\hat{p} - z_{a/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} < p < \hat{p} + z_{a/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}\right) = 1-a \tag{30}$$

2.3 Bayes' Bounding

According to Fukunaga, the estimation of a bound on the Bayes' error rate is accomplished by using a Leave-One-Out (L) and Resubstitution (R) analysis (described in the test below). The overriding concern in dividing up a finite set of data is maintaining independence between training and test sets. L, where just one data point is withheld from training and used for testing, is a special case of the more general holdout method where some fraction of the data is used for training and the rest for testing. As long as testing and training sets are independent, "the [holdout] and L methods are supposed to give very similar, if not identical, estimates of the classification error (19:221)."

Devijver and Kittler give an excellent discussion of these issues (13:343). They emphasize the fact that if one withholds most of a limited data set for training, leaving few for testing, one can have little confidence in the test. At the same time, if one reserves most of a limited data set for testing, it is not a good classifier design with respect to the statistics of the data. Also, Devijver and Kittler note that the hold out method tends to overestimate the actual error rate and gives cross-validation as a reasonable compromise in allocating an limited data set to training and testing subsets (13:355).

The disadvantage of the holdout method is that a tradeoff exists between the bias and variance of the estimated error rate. A full discussion of these effects is found in Fukunaga and Lachenbruch (27:1-11). Geman also discusses the impact of bias and variance on results (21). Bias reflects the accuracy of the classification results for a given experiment with respect to the accuracy that would be attained in a "real world problem." Vapnik gives a measure of the maximum deviation of estimated error rates from the true error applied to the real world (47). Jain paraphrases "Uncle Bernie's Rule," referring to Bernhard Widrow's rule of thumb that one needs about five to ten times as many samples per class as classifier free parameters in a pattern recognition problem (23:204). Finite data sets in high dimensional situations tend to be biased if the nature of the statistics is not captured by the data available. Over many runs, the estimates are unbiased if the estimates converge to the expected value. Variance relates to the consistency of the estimated classification rate over different sets of data.

A Bayes bounding analysis is implemented by Martin using k-Nearest Neighbors (k-NN), Parzen window, and neural network pdf estimates (32). In essence, the idea is to find an overly optimistic estimate of the Bayes error by training and testing on identical data sets (the R case). Likewise, the L method gives a slightly pessimistic estimate of the Bayes error. With respect to the Bayes error, the L value represents an upper bound. The premise behind the R method is that a classifier's performance can never be better than when the classifier sees the answers before answering the question. In the L method, if the data set contains M samples, M trials are made, sequentially withholding one test vector. In each trial, the classifier trains on $M - 1$ samples and tests on one sample. The total number of misclassifications across the M trials is used to estimate the error rate.

Martin's work varied the parameters associated with the pdf estimators (window size for Parzen, number of nearest neighbors for k-NN, and number of hidden nodes for neural networks) and adjusted the decision threshold accordingly. It had been found that the decision threshold plays a crucial role in the effectiveness of the Bayes error estimates (20). As summarized by Martin, when the L method is employed, the threshold should be adjusted to remove any influence the sample under test may have had in estimating the pdf. This

method for selecting the threshold is called "Option 2" by Martin and is utilized for the results generated in this thesis (33).

2.4 Feature Space Utilization

Efficient use of the feature space is an important element in engineering a pattern recognition problem. The desire to limit the dimensionality of the problem is driven by several concerns because high-dimensional data causes several problems in pattern recognition. Not only does the bias of estimates become significant (19:316-317), but also high dimensional situations require more sample points to adequately represent the statistics of the problem. The latter circumstance was explored in Foley (18) who studied the impact of using the same data set for both training and testing. Foley presents the ratio of samples per class to the number of features utilized as the key parameter. For, say, a quadratic classifier in a 192-dimensional problem, using a common training set and test set is not acceptable.

Another important issue is the "curse of dimensionality." It is possible to include too many features in a problem because the information added may be redundant information or noise (16:67). Jain points out that each added feature must increase the separation between classes by $\left(\frac{1}{n-d-3}\right)$ % (in terms of Mahalanobis distance) to avoid a roll-off in classification accuracy. In this equation, n is the number of samples and d is the dimensionality of the problem. Duda (16:77), citing Chandrasekaran (5), states that the inclusion of more and more dimensions will not affect performance if the features are truly statistically independent.

Foley cites other authors whose results emphasize the negative effects of limited sample size, including Cover (11), Estes (17), and Kanal (24). The underlying thrust of these articles is that the more information one has on the underlying behavior of the statistics of a random process, the better the performance will be. At the same time, one would like to find a minimal representation of the statistics of the data. When data is limited, which it usually is, one must develop techniques (such as the leave-one-out method) designed to estimate the performance of the classifier.

2.4.1 Projections and Transformations. When a researcher takes measurements, one is projecting a continuous natural phenomenon into a discrete representation in a feature space, R^N, where each feature represents a value along an axis in that space. This process can be visualized in the context of linear algebra, with the allowance that visualizing in 192 dimensions can be difficult. Together, the N axes form a basis set for R^N, which may or may not be linearly independent.

As an example, a continuous EM signal, such as a radar signature, may be projected into a feature space whose axes correspond to complex returns in range bins. Several items related to the physics of this problem bound it in a mathematical sense. Radar signatures are discrete, finite duration representations of physical processes and are thus finite energy. R^N is assumed to be a Hilbert space, which has the the attractive property of being a complete, inner product space. This property is important for several reasons. First, it means that R^N is a linear space and one may find linear manifolds (closed subspaces of R^N) in it. Also, it means that projections, via inner products, into those linear manifolds may be accomplished and that consistent distance metrics can be defined. The fact that R^N is a linear space implies rotations do not change the relative positions of points in that space.

Once a signal is projected and thereby vectorized into R^N, one can perform further linear projections and/or rotations to find efficiencies inherent to the data and classifier (19). An important strategy is to use the transformations to modify the coordinates of the feature space, exploiting those features which yield the most information about class separation (38:182). As long as the transformations are linear transformations, the underlying statistics are preserved.

Parsons focuses on two major methods for improving classification through transformations on R^N: decorrelating features (causing the correlation matrix to become non-zero only on the main diagonal) and maximizing separability. One method to decorrelate features is via the Karhunen-Loéve transform (KLT). The KLT essentially removes dependencies among linearly dependent features and forms a minimal representation of the data. Thus, R^N is transformed such that $R^N \rightarrow Q^N$, and Q^N possesses an orthogonal basis set. The steps for accomplishing the KLT are given in Parsons, mathematical details may be found in Fukunaga

(19:405-409). This linear transformation forms a new set of features which may be viewed as a minimal representation of the data in a mean square error sense. As pointed out later in this thesis, however, the KLT is not always the most successful with respect to classification error.

The essence of the KLT technique is to use the eigenvectors of the covariance matrix as an orthogonal basis of the new space. The technique yields an orthonormal basis in Q^N because eigenvectors are orthogonal and energy normalized. The corresponding magnitudes of the eigenvalues indicate the relative "importance" of each dimension. The beauty of the KLT is that the error in reconstruction produced by eliminating an eigenvector is directly related to the magnitude of the associated eigenvalue (19:410). Also, the KLT does not affect the underlying statistics of the problem because it is a linear transformation. Fukunaga extends this point by stating that the "class separability, for example the probability of error due to the Bayes classifier, is invariant under any nonsingular transformations (19:417)." This fact leads to the ability to use other rotations, as the one found in Lee's article, and projections, such as a multiresolution analysis.

This is all well and good, but to improve classification accuracy for a classifier, the real goal is to maximize the separability of the data. The KLT decorrelates features, but does not necessarily indicate which features may yield more information about class separation than others. The KLT does indicate in what directions the data as a whole tends to be spread. As pointed out by Parsons (38:185), the KLT finds a set of axes along directions of maximum variance which implies the directions of maximum separability. This strategy does not necessarily work in all cases (29, 38) but the door swings open on discriminant analysis.

Other feature discrimination techniques include the Fisher discriminant and the Fukunaga and Koontz method. Like the KLT, both of these methods seek out the dimensions in R^N that have the most variance. These techniques and several others are cited and briefly summarized in Lee (29:389). Lee points out that high dimensional data restricts the ability of these algorithms to perform in a computationally cost-effective manner. Also, if class means are close to one another, the results may not be meaningful.

2.4.2 Discrimination Using Decision Boundaries. As mentioned earlier in this chapter, a decision boundary $h(\underline{x})$, separates two classes in the N-dimensional feature space. For a quadratic Gaussian classifier, this equation is given analytically by combining Equations 18 and 24:

$$\begin{aligned} h(\underline{x}) &= .5(\underline{x}-\hat{\underline{\mu}}_1)^T\hat{\Sigma}_1^{-1}(\underline{x}-\hat{\underline{\mu}}_1) \\ &- .5(\underline{x}-\hat{\underline{\mu}}_2)^T\hat{\Sigma}_2^{-1}(\underline{x}-\hat{\underline{\mu}}_2) + .5\ln\frac{|\hat{\Sigma}_1|}{|\hat{\Sigma}_2|}. \end{aligned} \tag{31}$$

The decision is class ω_1 if $h(\underline{x}) < t$, where $t = 0$ for equiprobable *a prioris*. This equation represents an N-dimensional surface and is the foundation for all decisions in that pattern recognition problem.

A recent pair of articles by Lee and his co-author Landgrebe (28, 29) exploit this idea by transforming R^N based on the orientation of the most relevant part of the decision boundary. Their algorithm may be used to highlight more relevant features ("discriminantly relevant features") and leave out redundant features ("discriminantly redundant features"). The authors characterize this transformation as an improvement over the KLT with respect to classification. While the KLT minimizes mean square *reconstruction* error based on statistics of the feature vectors, it is not necessarily optimum in the sense of class separability. Lee's new approach is "based on the decision boundaries directly...and [predicting] the minimum number of features needed to achieve the same classification accuracy as in the original space for a given problem and [finding] the needed feature vectors (29:389)." That technique centers on extracting a basis set from the "effective decision boundary feature matrix" (EDBFM). The algorithm and its implementation are discussed next.

Lee uses the same notation as in this paper but adds a number of definitions to support the theorems and development of his argument. His key argument is to find a linear manifold of R^N called W^M. R^N is spanned by basis vectors β_i, $i = 1 \ldots N$, W^M is spanned by basis

vectors ϕ_j, $j = 1, 2, \ldots M$, and $M < N$. In W_M, the following equation holds for all $\underline{x}$:

$$(h(\underline{x}) - t)(h(\hat{\underline{x}}) - t) \;>\; 0. \tag{32}$$

where $\hat{\underline{x}}$ (M-dimensional) is the representation in W^M of $\underline{x}$ (N-dimensional). Lee states the physical meaning of this equation is that none of the decisions will change in the subspace (29:390). If the vector is correctly classified in both spaces then the product is always positive. If the vector is correctly classified in R^N and incorrectly classified in W^N, then the product will be negative. The problem is to find the basis vectors in the original space that are irrelevant to all decisions for all data points. In other words, one is seeking dimensions parallel to the decision boundary at all points. The following theorem lays the foundation for Lee's approach:

Theorem 1 *If a vector is parallel to the tangent hyperplane to the decision boundary at every point on the decision boundary for a pattern classification problem, then the vector contains no information useful in discriminating classes for the pattern classification problem, i.e., the vector is discriminantly redundant (29:391).*

He ends up concluding "the effectiveness of the basis vector is roughly proportional to the area of the decision boundary that has the same normal vector (29:392)." This leads directly to the mathematical equation:

$$\Sigma_{DBFM} \;=\; \frac{1}{K} \int_S \underline{N}(\underline{x}) \underline{N}^T(\underline{x}) p(\underline{x}) d\underline{x}. \tag{33}$$

In this equation, $\underline{N}$ represents the normal vector at point $\underline{x}$ to the decision boundary; S represents the decision boundary surface; and $K = \int_S p(\underline{x}) d\underline{x}$.

The advantage of the Lee technique becomes more clear when a new surface is formed: S', defined as the "effective decision boundary." S' represents that portion of the decision boundary which separates most of the exemplars in the problem. Often, a very complicated decision surface can be simplified to a linear equation. Basically, one is removing outliers

from the problem, betting that future outliers will occur infrequently. This truncated surface is used in Equation 33 to create the EDBFM.

The EDBFM holds information about the orientation of the decision boundary in R^N. The connection is that normal vectors to the decision boundary indicate the discriminantly relevant directions in R^N. The outer product of a normal vector with itself yields contributions to the EDBFM across the decision surface, S. An outer product, which forms an N by N dimensional matrix, may be viewed as an operator (36). For the EDBFM analysis, one takes the outer product of a column vector with itself:

$$\underline{\Upsilon} \cdot \underline{\Upsilon}^T \tag{34}$$

If this operates on some other vector, one writes:

$$\left(\underline{\Upsilon} \cdot \underline{\Upsilon}^T\right) \cdot \underline{x}, \tag{35}$$

but this can be reorganized as

$$\underline{\Upsilon} \cdot \left(\underline{\Upsilon}^T \cdot \underline{x}\right). \tag{36}$$

The term in the parenthesis is a familiar inner product and is the projection of $\underline{x}$ on $\underline{\Upsilon}$. This results is a scalar, leaving

$$\xi \underline{\Upsilon}, \tag{37}$$

where ξ is the scalar. Thus, an outer product provides a way to project one vector ($\underline{x}$) onto a second vector ($\underline{\Upsilon}$) and then orient along the second vector in R^N. Lee's algorithm says the EDBFM is an average of outer products from vectors produced along the decision boundary. Because they are normal to the decision boundary, these vectors form a correspondence with discriminantly important directions in the feature space.

The eigenvectors of the EDBFM give an orthonormal basis set for the space in which classes are best separated, with respect to the decision boundary itself. The dominant eigenvectors of the EDBFM are associated with the directions in which one is most likely to cross the decision boundary. The dominant eigenvalues may be used to indicate the most relevant eigenvectors, as in the KLT method described earlier.

Note that the process of taking outer products to form the EDBFM parallels the KLT, in which outer products are used to form the covariance matrix. Both matrices hold information about the direction of maximum variance of their constituent vectors. The eigenvectors of each provide the tool to transform the space into a reduced feature space. Thus, a transformation using the matrix of eigenvectors of the EDBFM orients the space with respect to the decision boundary.

See the appendices for sample problems and computer code. The steps used in implementing this procedure are listed here and in the Lee article (29:394).

1. Classify the training samples using estimated mean and covariance matrices and retain only those samples from each class that are correctly classified. This step insures that the equations used below are solvable. Also, apply a chi-square threshold test to each class to eliminate outliers within the class. Use the Mahalanobis distance metric. Note that the Mahalanobis distance does not include the bias term. For ω_1, apply steps 2 through 6, below.
2. Apply the chi-square threshold test of ω_1 to ω_2, again using a Mahalanobis distance. In other words, use only those samples in ω_2 which are relatively close to the mean of ω_1. Retain a predefined, minimum number of the closest samples from ω_2 if too few pass the threshold test. One may use a different threshold than in step one. Steps 1 and 2 limit the calculation to the *effective* DBFM.
3. For every sample of ω_1, $\underline{x}_1$, find its nearest neighbor in ω_2, $\underline{x}_2$. Essentially, at this point, one is drawing a line between the points. Because only correctly classified samples have been retained, this line intersects $h(\underline{x})$.

4. Because all retained samples should straddle the decision boundary, find the point P_i where the line connecting $\underline{x}_1$ and $\underline{x}_2$ intersects the decision boundary, $h(\underline{x})$. This step forms an estimate of the complete decision boundary.

5. Find the unit normal vector, $\underline{N}_i$, to the decision boundary at the point P_i. Here, the subscript i represents the i^{th} iteration through this list.

6. Repeat steps three through five for all K_1 samples of ω_1 and form an estimate of the effective decision boundary feature matrix, Σ^1_{EDBFM} from ω_1, where

$$\Sigma^1_{EDBFM} = \frac{1}{K_1}\sum_{i}^{K_1} \underline{N}_i \underline{N}_i^T. \tag{38}$$

7. Repeat steps two through six for ω_2.

8. Calculate the final EDBFM with

$$\Sigma_{EDBFM} = \Sigma^1_{EDBFM} + \Sigma^2_{EDBFM}. \tag{39}$$

The process is generalized to the multi-class case (with K classes) with

$$\Sigma_{EDBFM} = \sum_{k}^{K}\sum_{j,j\neq k}^{K} P(\omega_i)P(\omega_j)\Sigma^{kj}_{DBFM}. \tag{40}$$

Lee stresses that the theorems he develops hold for multiclass problems, with the above equation collating the class to class comparisons, weighted by their probabilities (29:395).

The implementation of these steps in this thesis follows his algorithm closely, but not exactly. Estimated parameters of the discrimination function, $h(\underline{x})$ were calculated from a distinct training set to maintain independence between training and test sets. Equations which find intersection points and normal vectors are provided by Lee (29:395). The computer code in Appendix C implements the steps listed above along with the equations used to find the intersection in Step 3 and the normal vector in Step 4. Essentially, the same results as found

in Lee's sample problems were produced with this code. The code in Appendix C is set up to reproduce Lee's sample problems (Examples 3, 4, and 5 (29:396-397)).

2.4.3 Wavelet Analysis. Wavelet decomposition in the context of a multiresolution analysis (MRA) is a relatively new technique which has especially matured in the last ten to fifteen years. The seminal article tying together the mathematics behind using wavelets in an MRA is Stephane Mallat's, *A Theory for Multiresolution Signal Decomposition: The Wavelet Representation* (31). The discrete wavelet transform projects a signal into a series of nested subspaces and their orthogonal complements. The analogy with Fourier series frequency representations, in particular windowed Fourier transforms, is tempting, but not quite exact. The wavelet kernal is parameterized with shifts and scale while the windowed Fourier analysis is parameterized with shifts and frequency. Instead of frequency as the key parameter in forming the transformation kernal, scale is the key parameter. With respect to the UHRR radar problem, wavelet analysis looks to pick out unique scale information (class to class) from the time-based signature.

The nested subspaces are called approximation levels. Projections into them are analogous to successive low pass filtering operations. See Mallat on how the filter coefficients are derived. In many cases, filter coefficients may be chosen tailored to the problem at hand. The resultant approximation signals have frequency content roughly corresponding to octave subbands. Referring to Figure 3, the approximations are labelled "A," "AA," and "AAA." The associated detail spaces contain information "lost" when one projects into an approximation space. Each approximation is a subspace of the original space and of the space of the approximation of the previous level.

Projections into the detail spaces result from high pass filtering of the approximation signals. The filter generating detail signals is derived from the filter generating the approximations and may be viewed as quadrature mirror filters (31). Each detail space is orthogonal to the approximation spaces and detail spaces at its level of scale and below.

During an MRA, filtering is applied recursively to the approximation signal. A direct sum of an approximation signal and detail signal is employed to reconstruct the approximation at the next higher level of scale. The intricacies of the mathematics may be found in Mallat's article. Note that with each projection, the signal is downsampled by 2. Each set of approximation and detail coefficients is a representation of the original signal in that subspace.

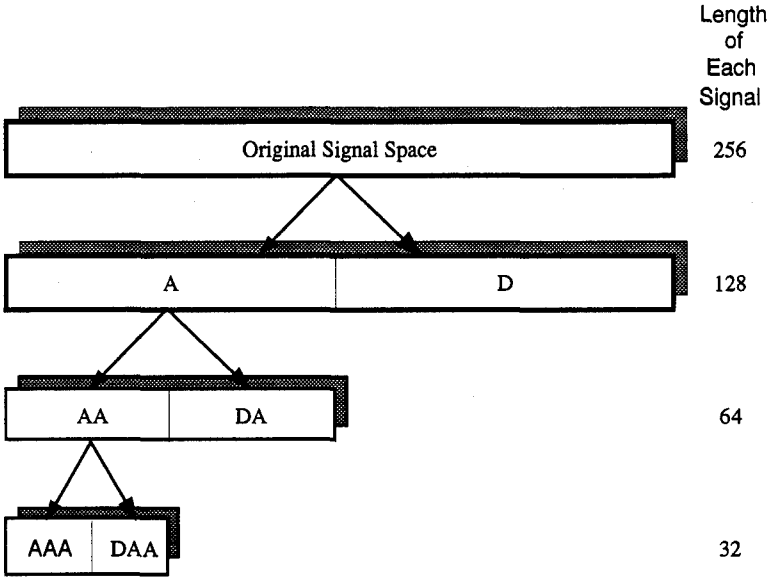


Figure 3. Original Signal Space Projected with Conventional Multiresolution Analysis

For the purposes of this thesis, wavelets provide a tool to extract relevant scale information about the targets. Wavelets will be used in the context of finding alternative orthonormal bases for the space in which the original signal is represented. The wavelet representations of the original signal are used as inputs to the Gaussian classifier.

Wavelet packets are a modified version of the conventional MRA in that detail signals are recursively filtered, as well. As pointed out by Coifman (10:714), a wavelet packet library "corresponds roughly to a covering of 'frequency' space." The term "covering" implies that the set of signals that can be represented in the original space can also be represented by the bases generated through the MRA. The wavelet packet bases are organized into nodes of a

binary tree, as shown in Figure 4. In the context of Section 2.4.1, representations of the signal are projected into subspaces at each level of decomposition in the tree. Instead of a standard MRA, in which only approximation signals are recursively decomposed, all detail signals are recursively decomposed at all leaves in the binary tree. The key in using wavelets instead of a windowed Fourier transform is that many of the detail spaces are orthonormal. This implies that the information contained in a detail signal is unique relative to other approximations and details at the same scale or below.

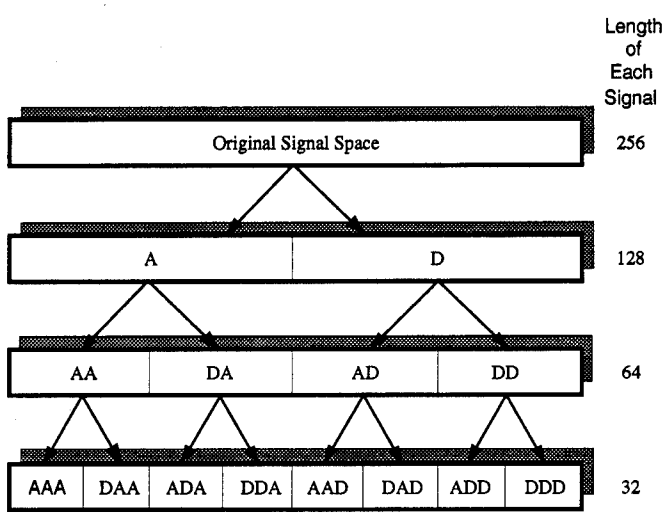


Figure 4. Original Signal Space Projected into Wavelet Packet Subspaces

For feature discrimination, the idea is to find those scales which contain the relevant information for classification. In the Coifman article, as implied by its title, "Entropy-Based Algorithms for Best Basis Selection," the premise is that entropy is an appropriate criteria in determining the best, minimal set of bases to represent a signal. This concept was applied by Chang and Kuo (6) with respect to identifying textured images. Their goal is to find the minimum number of features sufficient for classification (6:429). However, in the entropy measure, they are using a tool designed to give a minimal representation of the original signal, and is not specifically designed to minimize classification error. Recently, Coifman has developed a technique which selects scales based on classification error (9).

Chang's approach is to recognize that certain textures have little or no frequency information in certain bands. They "detect the significant frequency channels" by employing an averaged l_1 norm (6),

$$e(\underline{x}) = \frac{1}{N}\sum_{i=1}^{N}|x_i|, \tag{41}$$

where x is the vector of features at a given approximation or detail level.

The criterion for retaining a given decomposition level, or subspace, is to compare its energy to the largest energy value at that scale. Chang states if $e < C \cdot e_{\max}$, then stop decomposing at that level. In other words, this scale is relatively unimportant to the representation of the original signal. Chang goes on to lay out a training and classification scheme using the l_1 norms as features.

The essence of Chang's procedure is to calculate the "energy map" of a given texture or class during training. Complete tree-structured wavelet packet bases are computed for each training sample and l_1 norms are calculated for each set of approximations and details at each scale. All of these values are retained, no matter what the relative magnitudes of the energies may be. When a test vector is presented to the classifier, it is decomposed in the same way. This time, only those levels with significant energies are retained as features. These features are then compared to the corresponding features in each template and one's favorite measure of similarity (such as Equation 24) is applied.

Wavelets have great potential as a frequency analyzer, but one disadvantage is their shift variance. Because alignment and registration are critical with respect to the UHRR radar data, a way to address this issue must be found. A solution utilized in this thesis was presented by Suzuki in a report written as part of her PhD minor examination (46). In this technique, instead of downsampling, all values from a given filtering operation are retained at each level of decomposition. Thus, for a 256-bin radar signature, all approximations and details would also contain 256 bins. The shift variance problem is solved, because the downsampling does not remove every other range bin from consideration (46:9). Other solutions do exist,

including an article by DelMarco and Weiss (12) which applies wavelet packets and a shift invariant wavelet transform with damped sinusoids as test vectors.

One other important aspect of the multiresolution analysis must be addressed in the application of wavelets: edge effects. These will have a drastic impact on the features collected, especially as one progressively filters through each level of decomposition. Adaptive wavelets have been developed by Cohen, Daubechies, and Vial to alleviate the problem (8). These authors indicate more *ad hoc* techniques also exist which are easier to implement. Some examples are: periodically extending the signal and reflecting the signal about its endpoint to form a signal twice as long. Circular convolution during filtering may be employed to invoke the periodic extension. Sometimes, if the signal trails off to zero at the endpoints, zero padding is effective. Finding an effective strategy is often problem dependent.

Other researchers have used MRA's and wavelet packets for signal processing and, in particular, UHRR radar returns. A review of recent advances with respect to wavelet pattern recognizers is given by Benveniste and others (4). Chou and others use the multiscale analysis approach, noting that branches in the dyadic trees correspond to scale representations of a time signal (7). Baras and Wolk present a method for on-line automatic target recognition systems (3). Their proposed algorithms include vector quantization via aspect graphs and vector quantization techniques via the Linde-Buzo-Gray algorithm for clustering. Baras cites Gersho and Gray with respect to the the vector quantization techniques (22).

The work most similar to this thesis is a July, 1994 article by Coifman and Saito (9) which gives extensions to Coifman's 1992 article on entropy analysis of wavelet bases for minimum reconstruction error. In the July, 1994 article, classification error is addressed. The extension is they propose two techniques for feature selection from the wavelet packet bases. One technique is to use a Fisher discriminant on the wavelet packet bases. The other technique creates "an adaptive orthonormal basis [at each leaf in the tree] which minimizes a measure of the prediction error (such as l^2 error) for the regression problem (9:194)." The analysis in that article is restricted to synthetic data.

2.5 *The Adaptive Gaussian Classifier (AGC)*

The AGC takes UHRR radar signatures, preprocesses them and applies a Gaussian discriminant. The purpose of this section is to describe the algorithm outlined in Figure 5. This research makes an explicit distinction between Gaussian discriminants and the AGC. The term AGC always includes the entire preprocessing and alignment scheme as developed by Hughes Aircraft Corporation. The overall methodology is to first create templates from training data and then compare test signatures against them, as described earlier for the general pattern recognition problem. Each return in a range bin is treated as a dimension in the feature space. A template, as generated by the AGC, holds mean and variance information for each of the range bins for a given target class. An incoming, unknown signature is compared and aligned against class templates to find its "closest" match for classification.

2.5.1 The Data. UHRR radar waveforms are chirp radar returns from aircraft. Radar energy received from a target is demodulated with a linear frequency ramp function. Energy from scatterers at a given range is put into a given range bin. Each range bin contains energy corresponding to an incremental distance from the radar receiver. Radar energy corresponding to the nose of the aircraft is located in the lower-indexed range bins and energies from trailing edges are found in higher-indexed range bins. An important problem in this radar technique is a wrap around effect which occurs because of the range uncertainty imposed by the acquisition and demodulation process of the underlying chirp radar system. Details of chirp radar and radar stretching may be found in Stimson (44:217-223).

In this analysis, an unprocessed UHRR radar signature consists of 812 range bins. Each range bin has an in-phase and quadrature component. During acquisition, the signals have been oversampled by a factor of eight. Before classification or training, each radar signature is preprocessed. The specifics of ARTI Phase III data collection may be found in (1).

2.5.2 Preprocessing and Coarse Alignment. The preprocessing sequence is shown in Figure 6. The overall purpose of preprocessing is to normalize the data, correct the wrap-around problem, and align the data. Alignment occurs in a two-step process consisting of

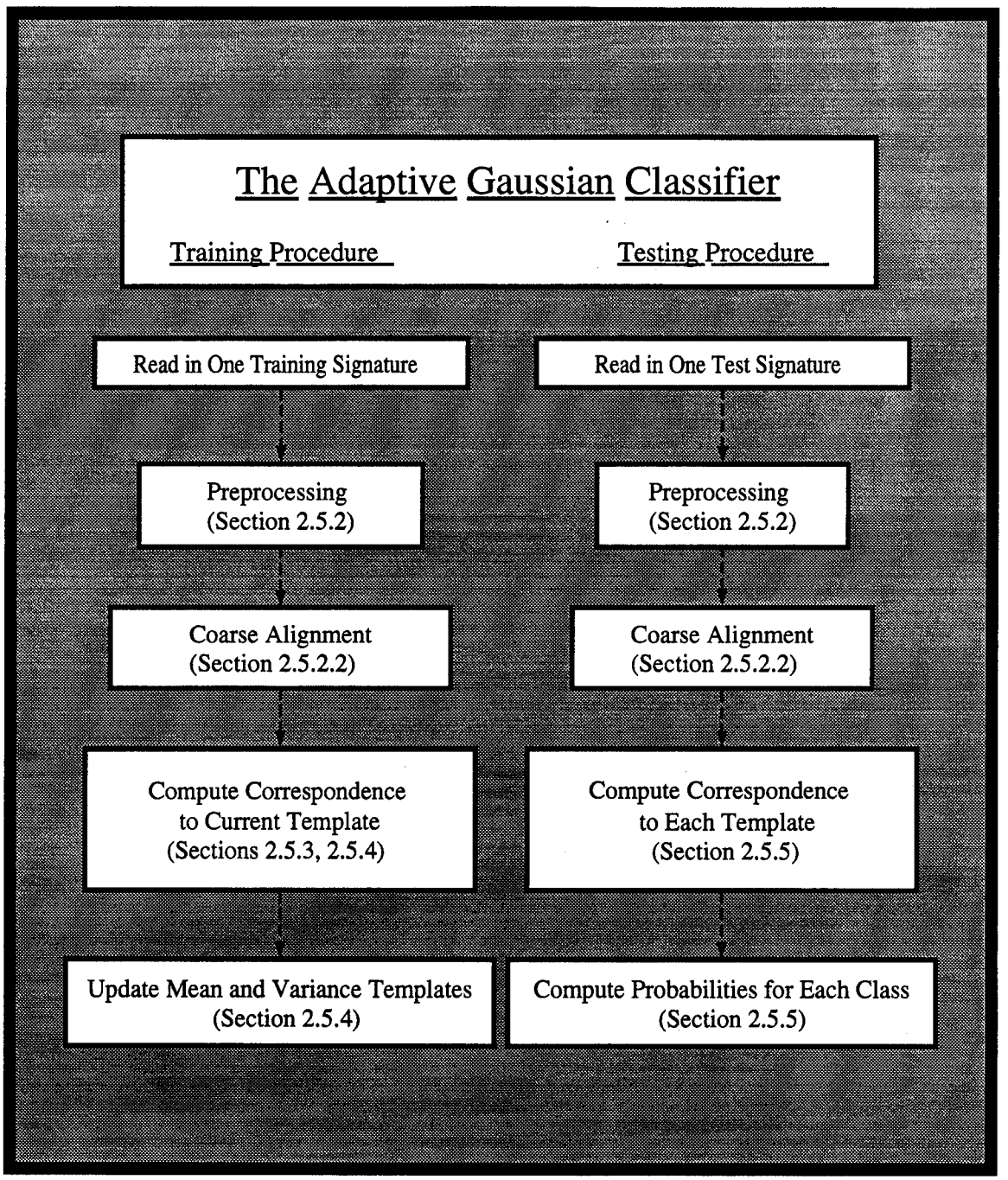


Figure 5. AGC Overview

coarse alignment and fine alignment. Technically, coarse alignment occurs in the preprocessing section of the AGC computer code. Fine alignment occurs as a separate step during template construction or classification.

2.5.2.1 Magnitude and Downsample. Let the stored UHRR radar signal be represented by the vector, $\underline{v}$, with the complex value of the return in the i^{th} range bin represented by $v[i]$. Each range bin contains in-phase and quadrature phase components $v_I[i]$ and $v_Q[i]$. The first step in preprocessing is to take a simple magnitude for all i:

$$v_{mag}[i] = \sqrt{v_I^2[i] + v_Q^2[i]}. \tag{42}$$

The resulting signature vector is downsampled by three, after deleting the first 22 bins and final 22 bins, to produce a new signal vector, $\underline{a}$. The assumption here is that the initial range bins are instrument noise associated with the acquisition process. The new signal vector $\underline{a}$ is of length 256.

2.5.2.2 Circular Centroid. A circular centroid (CC) performs the coarse alignment on $\underline{a}$ and solves the wrap-around problem. The CC finds the power centroid of a signal when the starting and ending points of the signal are assumed unknown. This coarse alignment of a signature is designed to get a given signature within ± 18 range bins of proper alignment with its class template. The number 18 was chosen as the range of shifts in experiments performed by DeWall (14). This limit on the window helps prevent confusion during testing because exemplars are less likely to be matched against the wrong template.

Mathematically, the CC operates on the signal vector $\underline{a}$ as follows:

$$y_{wgt} = \sum_{i=1}^{256} a[i] \cdot \sin i\Omega \tag{43}$$

$$x_{wgt} = \sum_{i=1}^{256} a[i] \cdot \cos i\Omega, \tag{44}$$

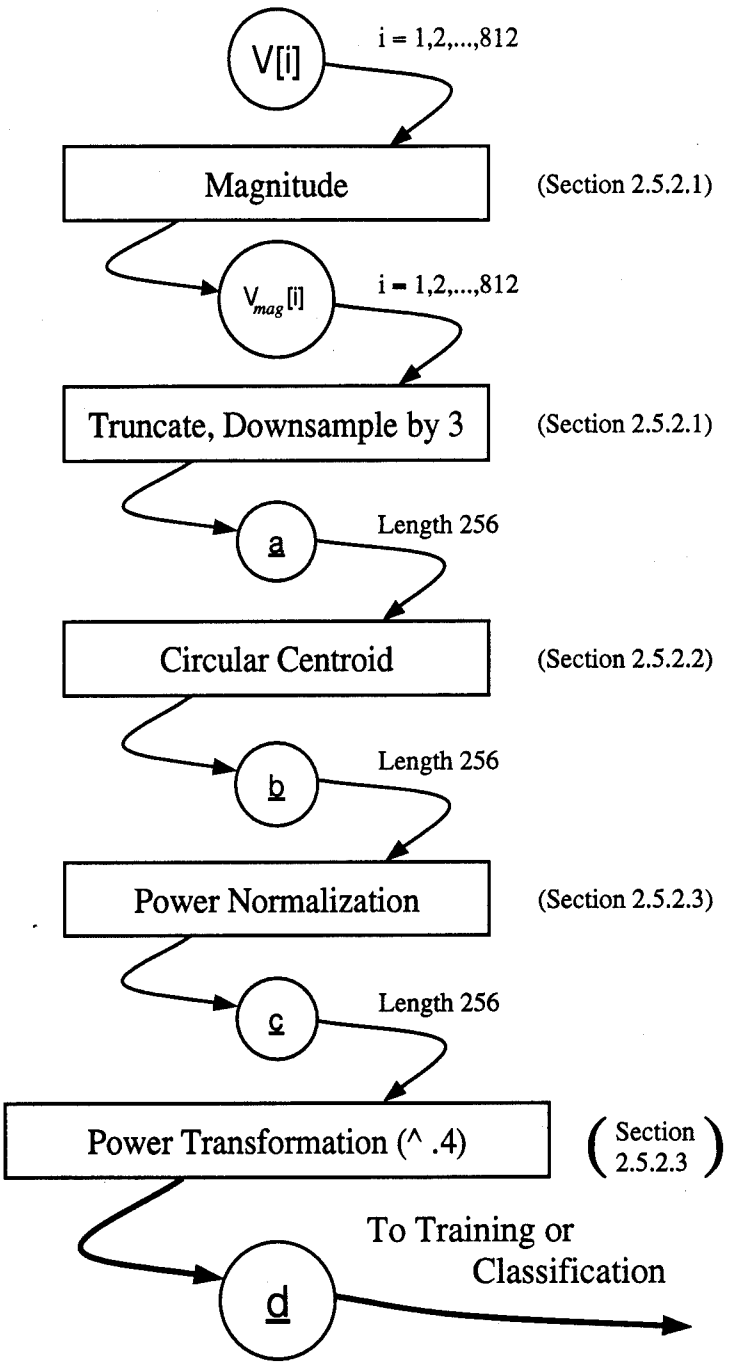


Figure 6. Preprocessing Steps of the AGC

where $\Omega = \frac{2\pi}{256}$. y_{wgt} and x_{wgt} represent the concentration of power along that axis in the xy-plane. These coordinates correspond to a point, β, as shown in Figure 7. The terms given in Equations 43 and 44 correspond to the real and imaginary components of the fundamental frequency in Fourier analysis. Thus, in a sense, the CC is a measurement of the energy content of the signature in a single frequency.

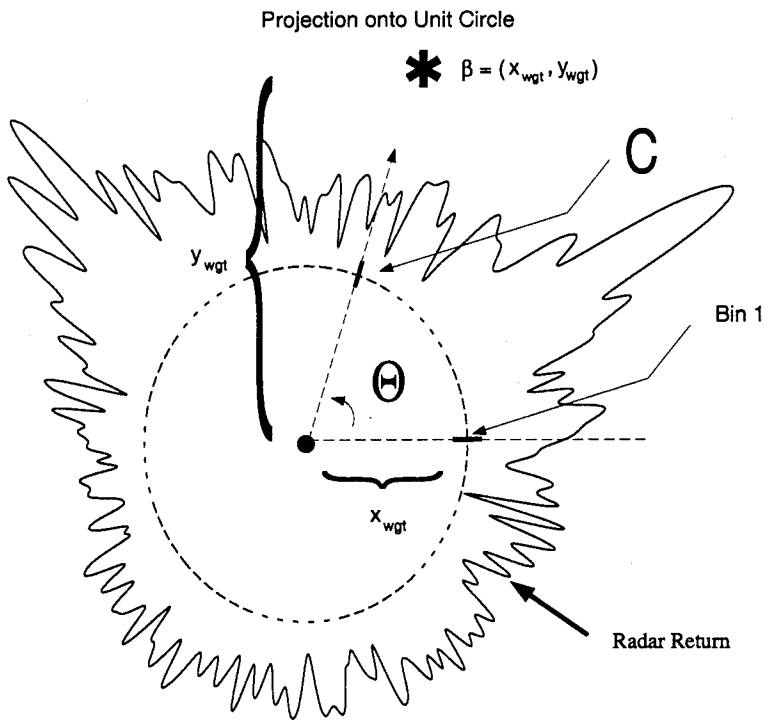


Figure 7. Circular Centroid

In Figure 7, to find the angle, Θ, corresponding to β, take the arctangent of its components:

$$\Theta = \arctan \frac{y_{wgt}}{x_{wgt}} \tag{45}$$

The center bin index, C, is found by converting back from radians to bin index number:

$$C = \frac{\Theta}{\Omega}. \tag{46}$$

Once this central bin index is found, it is a simple matter to rearrange the signal vector into a signal vector, $\underline{b}$, whose power centroid is its center bin.

The coarse alignment sequence is diagrammed in Figure 8. CC takes the signal vector and "wraps" it into a circle in 2-space with x and y components. Relative power strengths in the directions of the x and y axes are found and the coordinates of this point give the direction of the circular power centroid from the origin. The signal is then adjusted and "unwrapped" to complete the sequence.

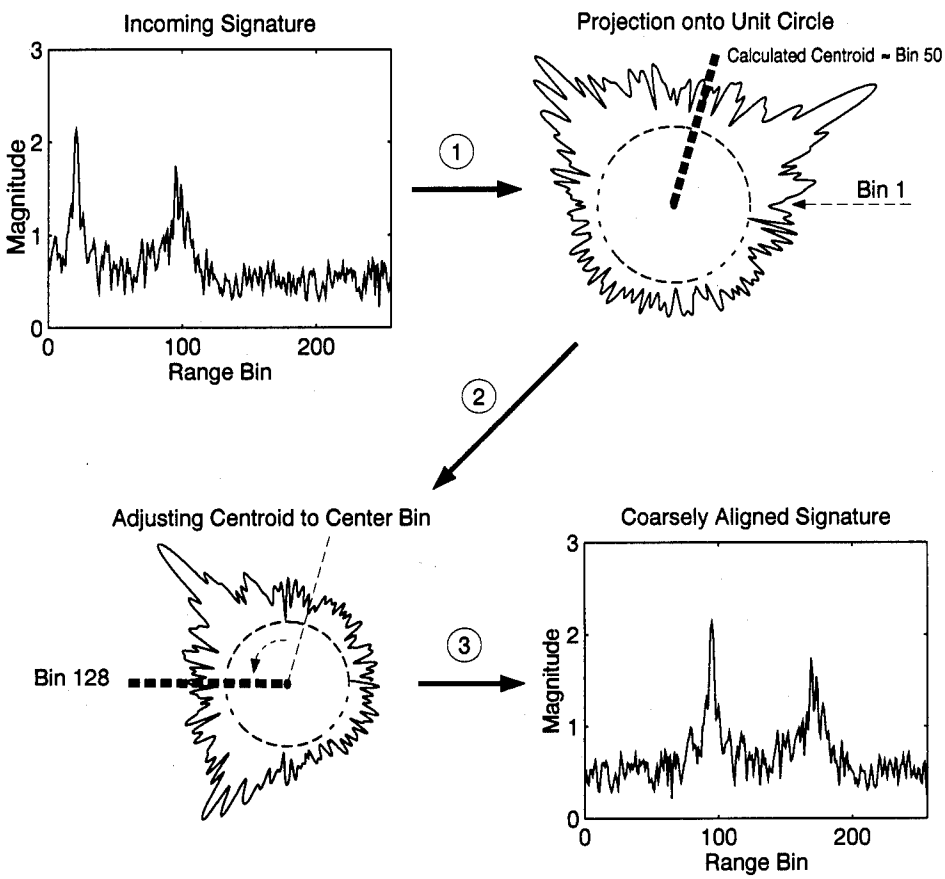


Figure 8. Coarse Alignment Process

The disadvantage of the coarse alignment technique is that abnormal flashes in the return of one exemplar of a given aircraft may cause improper alignment. These flashes can wash out information and cause an exemplar to appear very different from its class template,

especially in the sense of alignment. Such a flash problem is illustrated in Figure 9. This figure shows two signatures from one class. The signal with the flash no longer as similar with other members of its class. The noise flashes will affect the coarse alignment procedure and judgments of similarity during classification.

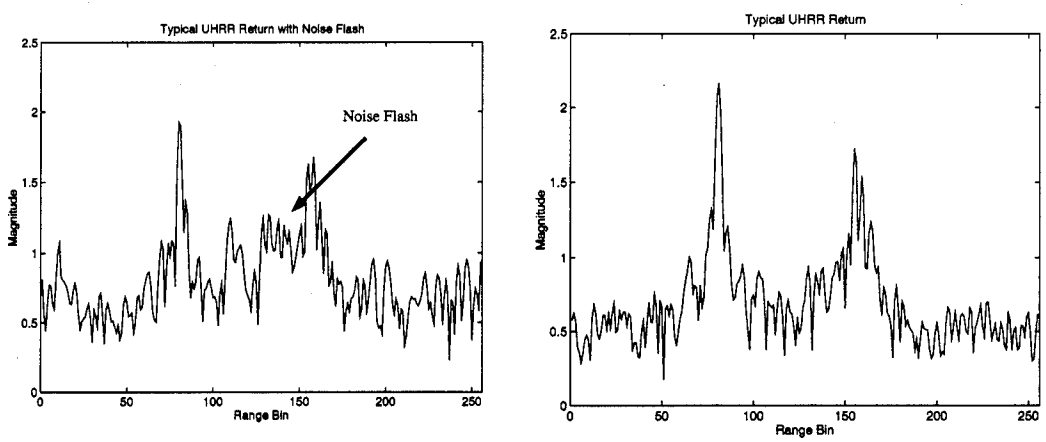


Figure 9. Left: Class **f2**, Signal "A," with Noise Flash; Right: Class **f2**,"B," Typical Signal

2.5.2.3 Power Normalization and Transformation. Each signal is energy normalized, as would be expected in any algorithm using correlations to align signatures against templates. Mathematically, compute the energy normalized version of the signals:

$$P = \sqrt{\frac{\sum_{i=1}^{256}(b[i])^2}{256}} \tag{47}$$

$$c[i] = \frac{b[i]}{P}. \tag{48}$$

The power transformation shown in Figure 6 makes the data more "Gaussian-like". The underlying distribution of the UHRR radar data is Rician (34), but the AGC inherently assumes a Gaussian pdf. The intention is to make the underlying UHRR radar pdf more closely appear Gaussian. This is a random variable transformation described in Fukunaga

(19:76). The transformation is:

$$d[i] = (c[i])^{0.4}. \tag{49}$$

The radar signature ready for training or testing is designated by the vector $\underline{d}$. This vector is incorporated into a template during training or tested against class templates during testing.

2.5.3 Training. The user of the AGC may choose how many of the 256 remaining range bins to use for training and testing. The algorithm leaves off bins symmetrically from each end of the signature. The default value is set at 192 range bins. Exemplars, denoted by $\underline{d}$ again, form 192-dimensional templates, where each range bin is a feature. The template file holds mean and variance information for each range bin. Template calculations are made sequentially, as each $\underline{d}$ is presented after preprocessing. Features are assumed to be uncorrelated.

Here, $\hat{\mu}_k[i]$ and $\hat{\sigma}_k^2[i]$ represent the mean and variance for the i^{th} range bin after k signatures have been presented for training. The equations implemented in the AGC's computer code are straight forward and follow from the theory presented earlier. The initial values are set from the first exemplar, $d_1[i]$:

$$\hat{\mu}_1[i] = d_1[i] \tag{50}$$

$$\hat{\sigma}_1^2[i] = 0. \tag{51}$$

For the k^{th} exemplar,

$$\hat{\mu}_k[i] = \frac{(k-1)\cdot\hat{\mu}_{k-1}[i] + d_k[i]}{k} \tag{52}$$

$$\hat{\sigma}_k^2[i] = \frac{(k-1)\cdot\hat{\sigma}_{k-1}^2[i]}{k} + \frac{\hat{\mu}_k^2[i]}{k-1}. \tag{53}$$

The latter equations, which recursively calculate mean and variance estimates are given by Schalkoff (41:65).

2.5.4 Adaptation. In the context of this thesis, the term "adaptive" only describes the process of shifting the incoming signal to the developing templates, which is formed iteratively as each exemplar is read during training. There exists code at WL/AARA which incorporates "adaptation" in another sense. In that case, a compensation parameter is added to the routine which accounts for changes in magnitude of the incoming signal (34). Thus, data taken at different ranges may be compared even though data from nearer ranges may have higher magnitudes than fainter returns. In the computer code itself, mean and variance calculations are made after finely aligning the k^{th} exemplar with the $(k^{th}-1)$ mean template. For training, alignment is found in the sense of a minimum Euclidean distance.

The exemplar is shifted until the minimum distance is found between the exemplar and mean template. Shifts are made up to eighteen bins in either direction, relying on the premise that the CC has already coarsely aligned the training signature to the class mean template. When the CC fails to place the "true" centroid within 18 range bins of its class mean, this fine alignment step fails. Shifts are made circularly: in other words, bins shifted off one end reappear at the other.

The mathematics are as follows, with $D_k[j]$ representing the squared Euclidean distance for the k^{th} exemplar offset by j:

$$D_k[j] = \sum_{i=1}^{256} \left(d_k[(i+j) \odot 256] - \hat{\mu}_{k-1}[i]\right)^2 \tag{54}$$

$$j \in [-18, 18], \text{(j an integer)}. \tag{55}$$

The $\odot$ symbol represents the modulo operator and implements the circular shifting of the exemplar, $\underline{d}$. From the above equation, the j^{th}_{best} shift is chosen that minimizes D_k. Then, instead of Equation 52,

$$\hat{\mu}_k[i] = \frac{(k-1)\cdot\hat{\mu}_{k-1}[i] + d_k((i+j_{\text{best}}) \odot (256))}{k}. \tag{56}$$

is used. This equation uses the best matched version of d_k in a Euclidean distance sense. The calculation for $\sigma_k^2[i]$ remains the same.

The above steps are repeated for as many exemplars as the researcher wishes to use to form the class template. The process is executed separately for each target class, and presumes the use of labelled data. Templates generated during training are placed in separate target files. For testing, individual templates are concatenated into a single pattern file.

2.5.5 Testing. Noting Figure 5, for each unknown test signature presented to the classifier, the preprocessing steps outlined in section 2.5.2 are performed to generate a normalized and coarsely aligned feature vector.

The testing algorithm computes Mahalanobis distances between the unknown vector, $\underline{d}$, and each class template vector contained in the pattern file. As in training, the test vector is circularly shifted against each class template for fine alignment and the best (minimum) value is stored for each class. The AGC implements a portion of the discriminant equation, Equation 31. Only half the terms are used, as shown here, for a comparison with the ω_1 template:

$$m_d(\underline{x}) \quad = \quad 1./(\underline{\hat{\sigma}}_1^2).*(\underline{x}-\underline{\hat{\mu}}_1)\cdot(\underline{x}-\underline{\hat{\mu}}_1)^T, \tag{57}$$

where $\underline{\hat{\sigma}}_1^2$ is a row vector containing estimated variances for Class . The ".*" operator indicates element by element multiplication and the "./" operator indicates element by element division. The equation has simplified because of the assumption of uncorrelated features and because one is calculating the *a posteriori* probability for a specific class. The linear algebra works out the same as with a full covariance matrix, but computation time is saved. It is at this point that two important assumptions in this classifier become evident. Features are assumed to be uncorrelated (34) and the decision boundaries are assumed to be quadratic. That is, covariance matrices are estimated for each class and they are non-zero only on the main diagonal.

The testing process is the same as training, except that the Mahalanobis distance is used *plus* the bias term, for example, B_1 for class 1. The bias term for class 1, meaning the

determinant of the covariance matrix simplifies to

$$\left|\hat{\Sigma}_1\right| = \prod_{i=1}^{256} \hat{\sigma}^2[i] \tag{58}$$

$$B_1 = \sum_{i=1}^{256} \ln \hat{\sigma}^2[i]. \tag{59}$$

For a given test signature, $p(\omega_1|\underline{x}) = m_d + B_1$ is calculated for each template and the values are put in a matrix and saved to an output file. This file contains K columns with respective classification distances to each class template. The user must use his own algorithm to translate the results to a confusion matrix.

2.6 Summary

This chapter provides the tools used in the remainder of this thesis. The statistical theory is given which allows for an evaluation of the AGC by estimating the AGC's classification rate. Methods for understanding the data with respect to separability issues and feature discrimination are provided. The effects of assuming that the underlying distributions are Gaussian may now be evaluated. Feature discrimination using the technique given by Lee and Landgrebe is applied and extended. Also, the groundwork for using a multiresolution analysis and discriminating wavelet features is provided. The next chapter explains the methodology used to meet the thesis' goals.

III. Evaluation Methodology

3.1 Introduction

The purpose of these experiments is to evaluate the impact of alignment and radar signatures with flashes on the AGC. These tests attempt to determine whether there is a statistically significant problem with respect to alignment. The AGC is a simple Gaussian classifier with a couple of twists. The first twist is the alignment procedure, which is divided into two parts. The first part is the circular centroid technique described in Chapter 2 to "coarsely align" data. The other part is the fine alignment procedure, which is an integral part of the final decision making process. This is because each test signature is ideally aligned against each class template independently and then the best "match" among the templates is declared the winner. This process is known as establishing proper correspondence between an exemplar and template. The second twist in the AGC is to calculate the mean and variance of the features recursively during training. After being circularly centroided, each training signature is finely aligned against the cumulative mean template and is then added to the mean and variance templates proportionally. This twist may have an impact on results because irregular signatures presented to the AGC early in the training process may adversely affect performance. One way to combat this problem is to repeat trials many times and randomize presentation order.

Three potential problems are evident with the implementation of the AGC. One is its ability to perform the coarse alignment. The alignment problem has been suspected as the source of most errors in UHRR classification (34). Noise flashes can cause the centroid of an exemplar to be far from the average centroid of others in its class. The circular centroid must adjust the signal to within ± 18 range bins to have a chance for correct registration with its actual class template during the fine alignment stage in testing. During AGC evaluation, this problem is addressed by removing problem signatures (signatures that are coarsely aligned such that the centroid is not inside the ± 18 window as required by the fine alignment process) and running the AGC with and without these signatures.

Another problem relates to the assumption of Gaussian pdf's. Not only does the AGC make an important assumption about the underlying statistics of the data, but as a quadratic classifier, it places severe conditions on the classifier because of the effective number of features which are used. During AGC evaluation, this problem is addressed by giving the same data sets to a non-parametric classifiers (k-NN and neural network). In this case, the alignment problem is not addressed, only the the ability of the AGC is compared to other classifiers.

The third potential problem rests with the data itself. When a signature is aligned poorly by the circular centroid, is there something inconsistent in that sample with respect to other samples of its class? This question has to do with the underlying information content of the signatures. During AGC evaluation, this problem is addressed by quantifying the underlying information content of the data by using a Bayes bounding experiment implemented by Martin (33).

The last portion of the chapter explores the feature discrimination technique developed by Lee (29). This technique seeks to find the most discriminantly relevant features. This research explores the AGC's performance with respect to these questions with computer resources available at AFIT. In Appendix B, the results for two alternative training schemes are given which yield moderately better results with respect to the alignment challenge.

3.2 Data Preparation

Data is parsed into three sets: "arbitrary data," "hand-aligned data," and "pristine data." The total number of signatures retained in each case is shown in Table 1. The criteria for separating the data are listed below.

arbitrary data This case includes all data from the original target set.

hand-aligned data This case includes all data from the original target set. Each signature is visually checked after the circular-centroid for proper alignment. If a signature does not fall within the ± 18 window after coarse alignment, then it is designated as an improperly

Table 1. Number of Signatures in Data Sets

Class	Pristine	Hand Aligned	Arbitrary
9a	986	1008	1008
aa	1006	1008	1008
f2	591	1031	1031
fa	998	1008	1008

aligned signature. Improperly aligned signatures only, referred to as "bad" signatures, are forced to fall within the ± 18 window by adjusting the position of the signal by hand. This was accomplished with an interactive algorithm developed for WL/AARA by Veda, Inc. When this data is submitted to the AGC for training or classification, the circular centroid subroutine is turned off to retain the human aligned positionings of the bad signatures.

pristine data This case eliminates all signatures designated as bad from the database. The pristine case retains only those signatures which the CC subroutine successfully placed within the ± 18 window. Class **f2** has the greatest number of problem signatures, while class **aa** has only two problem signatures.

3.3 Impact of Alignment on Classification

3.3.1 Test Criteria. For each data set case (pristine, hand-aligned, arbitrary), the AGC is used in a four class environment. The performance metric used is a confusion matrix and the overall probability of correct classification P_{cc} is computed by averaging individual class classifications (19:66). Multiple trials bound P_{cc} with 97.5% confidence intervals. As stated earlier, these tests attempt to determine whether there is a statistically significant problem with respect to alignment.

3.3.2 Test Methodology. As summarized in Figure 10, three cases are tested:

Pristine This case simulates a situation where there are no mis-aligned signatures (that is, all signatures fall within the ± 18 window after the circular centroid). There are no coarse

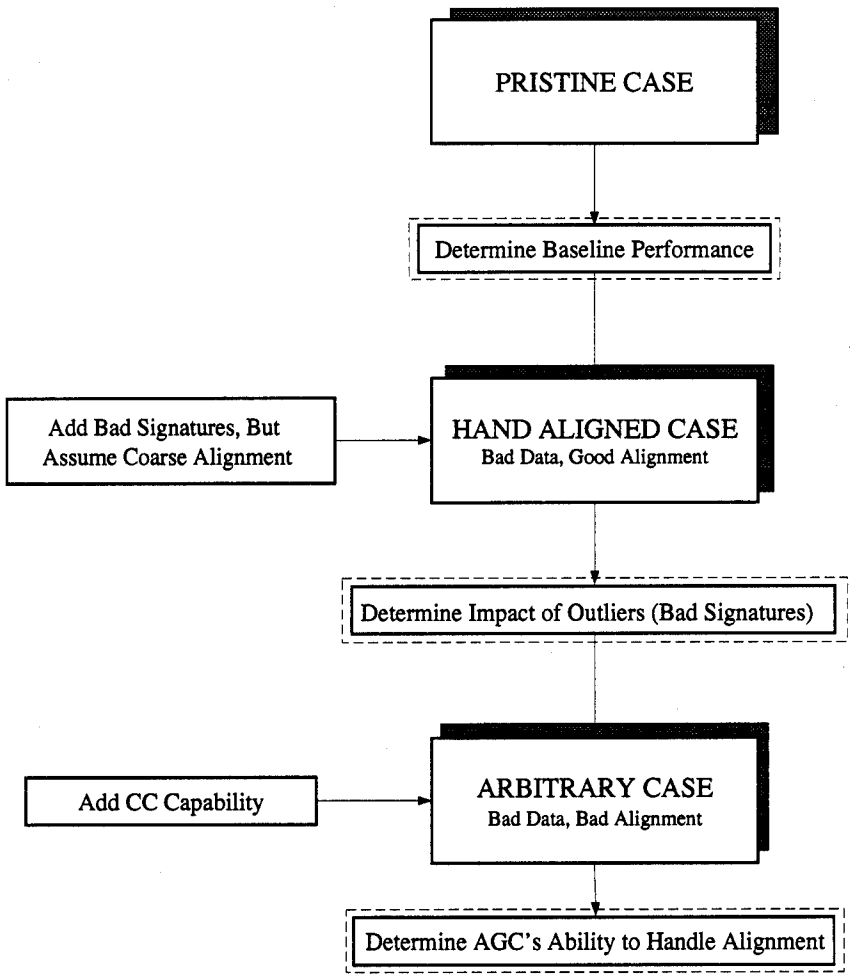


Figure 10. AGC Testing Methodology (See Section 3.3.2)

alignment problems and no problem signatures. In a sense, this removes alignment issues from the classification problem.

Hand-Aligned This case adds the bad signatures to the problem. Usually, bad signatures are observed to have noise flashes which cause the coarse alignment to fail (recall Figure 9). These bad signatures do not match the overall energy characteristics of the rest of their class, but are forced into the ± 18 window by human alignment. Thus, coarse alignment is performed by human perception and the CC within the AGC is turned off. In this case, there are no coarse alignment problems, but there are bad signatures. This case tests the ability of the classifier to handle outliers within classes.

Arbitrary This case allows the AGC system (circular centroid) to perform the coarse alignment on all available signatures. Both coarse alignment and bad signatures affect P_{cc}. This case tests the performance under the harshest of the alignment conditions.

Pristine vs. Arbitrary This case trains on pristine data only, but includes bad signatures in the test set. This case tests whether arbitrary data corrupts the template being generated to some extent.

In all of the cases, the fine alignment process remains the same.

3.3.3 Implementation. Trials are made using the hold-out method. Half the data set is used in training and half the data set is used in testing. Matlab code randomly splits the data into the training and testing sets for independent trials. These trials are independent in the sense that, for each trial, training and test sets are mutually exclusive. Trial to trial, there will be common vectors in the testing group. This fact does bias the results, because the same test vectors may be repetitively misclassified. For this thesis, the desire was to produce some confidence intervals on the results and the data layout was well suited to the randomization procedure devised. Also, the statistics of the classes seem to be well represented by the data available. The use of many Monte Carlo repetitions is designed to bound the final error rates and reduce the impact of presentation order to the training portion of the AGC. Leave one out results are given in Appendix B.

The randomly generated training and test sets are given to the AGC for computation, and Matlab code is used to interpret the output for classification. The AGC produces a matrix of distances where each row corresponds to a test vector and each column is its distance, or test statistic, with respect to a given class. Matlab selects the minimum value from each row as the "winning" class assignment. In all cases, *a priori* probabilities are assumed to be equal. Trials continue until statistical significance between overall P_{cc}'s for the pristine and arbitrary cases are established. This takes on the order of 1750 trials per case. The hand-aligned trials were repeated the order of 750 trials. Results from the trials are averaged together to produce final confusion matrices. All runs are completed on Sun Workstations.

3.4 Impact of Classifier Type

LNKnet implemented the non-parametric classifier comparisons. Trials were executed with respect to a *k*-NN and neural network. Because alignment was not being tested, the classifier was given signatures finely aligned with the actual class only. This biases the result in favor of correct classification, but still gives a relative measure of classification capability for this high dimensional data set. For comparison, runs were also made against a Gaussian classifier (implemented on LNKnet).

3.4.1 Data Preparation. Extracting features from the AGC was accomplished by going into the C code and writing signatures directly to a binary file just before inclusion in the class template during training. Thus, all alignment (both coarse and fine) and preprocessing steps are accomplished before extracting the signature from the AGC. Matlab was used to read the file and convert it to ASCII, the required format for LNKnet.

3.4.2 Test Criteria. For these tests, the same test criteria as Section 3.3 apply. Confusion matrices and 97.5% confidence bounds are generated.

3.4.3 Test Methodology. Test methodology is the same as above, but includes only the arbitrary and pristine cases.

3.4.4 Implementation. Runs are repeated in the same way as Section 3.3, with the exception that N-fold cross validation is utilized. This test procedure divides all data into equal subsets and then iteratively tests on one subset at a time, while training on the rest. The concept is the same as with leave one out evaluation, but with larger test sets. In this thesis, 50-fold cross validation is used to form confusion matrices as a measure of classification capability as described earlier. The number of trials available for execution was limited by LNKnet. This is because LNKnet produces massive parameter files which are directly proportional to the size and dimensionality of the data.

For the *k*-NN runs, 15 nearest neighbors are used. This number was selected because in the Bayes experiments below, 15 was generally found to be a value that produced the best results. For the Gaussian runs, LNKnet was set up as the AGC, with class dependent, uncorrelated covariance matrices.

The neural network runs used 192 input nodes, 25 hidden nodes in one layer, and 4 output nodes. Most of the standard settings within LNKnet were retained for the neural network runs. This included a step size of .1, a momentum of .6, a standard sigmoid function in the nodes, and the square-error cost function. Random presentation order was turned "on" and 50 epochs were used during training.

For each classifier type, equal class *a prioris* are assumed.

3.5 Impact of Data

This experiment is conducted according to Martin (33) and bounds the best possible classification rates one can hope for with the data at hand. In other words, it is a measure of the separability of the data. Martin's algorithm is designed only for two class problems. AGC runs were accomplished for all two class combinations in the same way as Section 3.3 for comparison.

3.5.1 Data Preparation. Data was prepared as in the previous section. Like LNKnet, Martin's code expects ASCII formatted data.

3.5.2 Test Criteria. Martin's code produces graphs showing the error rates generated using the L and R methods described in Chapter 2. Classification results are expected to be relatively high if the data is separable on a class to class basis.

3.5.3 Test Methodology. Methodology followed the procedure found in Martin's thesis (32), using only k-NN evaluation because it provides the most consistent estimates of the Bayes error. For the k-NN, the parameter k was varied to give a range of error rates as a function of k. Results were generated by taking the best case from the range of k allowed by Martin's code. As mentioned earlier, $k = 15$ was usually the best case.

3.5.4 Implementation. Martin's code performed 10-fold cross validation during testing and produces error rates with respect to varying the k parameter. Option 2 was used to determine the decision threshold in the L case. The results given in the next chapter were generated by taking the midpoint between the R and L case as the estimate of the Bayes error. For comparison, class to class runs with the AGC itself were made using the pristine and arbitrary cases.

3.6 Summary of AGC Methodology

The first part of this chapter describes the methodology used to evaluate the performance of the AGC. The overall thrust of the first section is to analyze preprocessing effectiveness. The idea is to begin with pristine data and add in one of two potential problems at a time. The first step includes problem data (bad signatures) to see whether it is something in the data which causes the fine alignment and/or subsequent classification to fail. The next step adds back coarse alignment as an issue. With the arbitrary case, the machine is allowed to do the coarse alignment. This experiment tests the effect the CC is having on the AGC. The overall thrust of the second section of this chapter is to analyze the modality of the data: Is classification hurt by limitations of a Gaussian quadratic discriminant? Alignment is removed as an issue. The third section addresses the separability of the data on a class to class basis. Results are presented in Chapter IV.

3.7 Discriminant Analysis of the UHRR Feature Space

This portion of the thesis explores the use of Lee's algorithm for feature discrimination with the UHRR problem. After validating the performance of the algorithm with simple example problems given by Lee, the algorithm was applied to a number of cases of the UHRR problem. Two of the validation examples appear in Appendix A.

All preprocessing was accomplished with Matlab code emulating the AGC algorithm, including alignment, template construction, and the generation of test statistics. Data was converted to Matlab formatted binary files and then classified with Matlab code. The Matlab code was verified by testing its classification accuracy against the AGC, and was found to perform the same for given data sets.

The results presented in Chapter 4 use 256 range bins because of the eventual extension to wavelet analysis, but similar results were observed for 192 range bins. The algorithm was run in the four class case for pristine data. Several two class situations were also tested involving selected targets.

The test sequence proceeds as follows:

1. During training, save template files for use as estimates of class means and variances.
2. Classify the test data.
3. Separate correctly classified signatures into a new data set for use with the Lee algorithm.
4. Provide the Lee algorithm with results from Steps 1 and 3.
5. Generate and save the EDBFM, its eigenvectors, and its eigenvalues.
6. Reclassify the data using a specified number of features.

Lee's article clearly implies that reclassification occurs in the new, transformed "EDBFM space" (29:397). Thus, one selects the number of features used in the new space and forms a transformation matrix composed of the eigenvectors corresponding to the eigenvalues of greatest magnitude. This selection process is the same as the ordering that takes place in a KLT. For this thesis, signature vectors are row vectors and the transformation matrix's

columns are filled with eigenvectors. To generate sample vectors in the reduced feature space, one post-multiplies a matrix of UHRR signatures by the transformation matrix. This post-multiplication conveniently executes the inner products required to perform the projections onto the new basis set.

Transformed training and test sets are used during "re-"classification. Both correctly classified sample points and incorrectly classified points are used during final testing. Dominant eigenvectors are added to the transformation matrix over several runs to get a feel for how many features are required to maintain the original classification rate.

At this point, an important caveat to the Lee algorithm becomes evident. When projecting into the EDBFM space, one is using linear combinations of *all* the original features. Thus, it seems, there is no advantage to reducing the *overall* dimensionality of the problem. One may infer from Lee's article that the dominant eigenvectors correspond to the dominant features (29:396). Also, one may infer that one is reducing the original feature space, but the technique is actually performing a projection into a space with fewer features (dimensions) (29:389,391).

An attempt was made to interpret the EDBFM's dominant eigenvectors in order to infer where the discriminantly relevant information lies in the *original* space. Referring to the 3-dimensional problem in Appendix A, one sees that the primary eigenvector's most significant components correspond to the discriminantly relevant features in the original space (*i.e.* the first two dimensions). This interpretation has an intuitive appeal because in performing an inner product, those components that are "most important" to the transformation will be those components that are weighted heaviest in the inner product. For example, in the case of projecting a two dimensional vector onto the x-axis, the y-component of the vector plays no role in the projection. In more complicated situations, where there are non-zero elements of the projection operator being ignored, one is throwing away some information about how the transformation occurs.

The most dominant eigenvectors and the least significant eigenvectors tend to "point to" very different feature elements. Again, these eigenvectors are ordered with respect to the magnitude of the corresponding eigenvalue. Figure 11 shows the disparity.

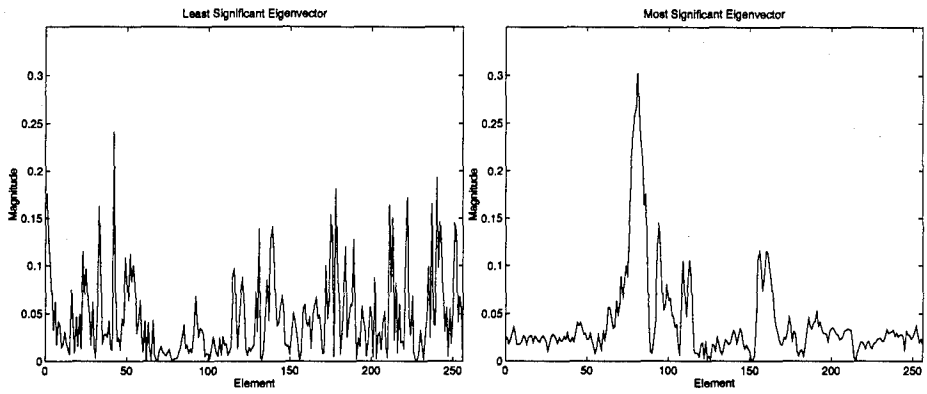


Figure 11. Comparing Eigenvectors: 4 Class, Pristine Data

The key point in this analysis, however, is that the dominant eigenvectors may provide information about the discrimination power of features in the original space. In Figure 11, features 80–90, 105–115, and 150–160 would be interpreted as important to discrimination. This idea is tested by using dominant eigenvectors to choose features in the original space and then re-classifying. This technique is successfully applied in Kocur's work on breast cancer detection (25). As above, the most significant eigenvectors direct one's attention to specific features. Her choices for relevant features, based on anlysis of the EDBFM agree with Steppe's analysis of feature saliency with respect to neural networks (43).

Features are selected by averaging the ten most significant eigenvectors and choosing the elements with significant magnitudes as corresponding to relevant features. The selection of ten eigenvectors had the best classification among the options of choosing the best eigenvector only or averaging the five most significant features. Various numbers of features are used and tabulated.

3.8 Summary

After describing methodology for the underlying pattern recognition problem, this portion of the thesis outlines a feature discrimination technique for the UHRR radar problem. The advantage of Lee's approach is to base feature saliency on classification error instead of reconstruction error. Lee finds a rotated space in which a reduced number of features are required to attain about the same classification rate as the original space. The point addressed in this thesis is that one may attain true feature reduction by interpreting the eigenvectors of the EDBFM. Results comparing the performance of the feature discrimination approaches are given in Chapter 4.

IV. Results

4.1 Introduction

This chapter presents and summarizes results from the previous chapter. Results are presented in the form of confusion matrices for the full, four-class comparisons. 97.5% confidence intervals are given for all estimates of the classification rates. For two-class comparisons, results are summarized in aggregated tables.

4.2 Impact of Alignment

Table 2 shows overall classification performance and the next three tables show the interclass details. Several observations and conclusions may be made from each of the tables. Overall, Table 2 shows there is a statistically significant alignment problem in the four class case. The average P_{cc}'s presented here were observed to be stable and more trials (on the order of thousands) would show statistical separation even for the hand aligned case.

Table 2. AGC: Overall P_{cc}

Case	P_{cc} (%)	97.5 % Confidence Interval (%)
Pristine	90.16	±1.31
Hand Aligned	88.66	±2.76
Arbitrary	87.14	±1.59
Pristine vs. Arbitrary	82.60	±3.32

The results presented in this section show that the "bad" data, represented by the hand-aligned row, affects classification to some extent and the inability of the circular centroid to coarsely align the bad data affects classification to an approximately equal extent. The two factors combine to yield a three percent degradation in performance in the overall, 4-class case. The impact of alignment and data problems, however, are class variant as shown by the individual rows of the confusion matrices.

Recalling Table 1, the AGC has the most difficulty with class **f2** with respect to alignment because it had the highest percentage of bad signatures. As shown in Tables 3 through 6, the AGC's ability to classify members of **f2** degrades by ten percent from the pristine case to the arbitrary case. Similarly, class **fa** degrades by about four percent. The performance of the other two classes remains relatively constant across the three cases, with **9a** always performing poorly and **aa** always performing rather well.

This result implies that the alignment problem can be extremely data dependent. The fact that class **9a** has poor performance across all three cases says that the class may have more than one cluster center in the feature space. Here, the unimodal assumption of the Gaussian classifier hurts performance. The fact that class **aa** performs well across all three cases is because there are very few samples characterized as bad and that it is well separated from the other classes. Measures like the Fisher discriminant would probably work well with this class, and perhaps the **f2** and **fa** since they give very good results. However, the issue that will always play a role in any such analysis is how one has decided to make alignment decisions. Classes **f2** and **fa** have the most bad signatures the classification rates degrade sharply, as would be expected.

In the case of training on pristine data and testing on arbitrary data, note that the AGC appears capable of "learning" the misalignments induced by bad data. This is shown by comparing Table 6 with the other tables.

Table 3. AGC: Pristine Data

Actual Class	Assigned Class				P_{cc} (%)	97.5 % Confidence Interval (%)
	9a	aa	f2	fa		
9a	374.03	49.58	1.55	67.84	75.87	±1.89
aa	.572	499.11	0.00	3.32	99.03	±0.43
f2	.038	2.96	285.32	6.68	96.72	±0.77
fa	8.92	30.49	3.27	456.32	91.44	±1.67

Table 4. AGC: Hand Aligned Data

Actual Class	Assigned Class				P_{cc} (%)	97.5 % Confidence Interval (%)
	9a	aa	f2	fa		
9a	372.76	58.74	1.56	70.94	73.96	±3.82
aa	2.25	495.8	0.01	5.94	98.72	±0.01
f2	4.46	12.47	484.78	13.29	94.13	±2.04
fa	14.97	43.46	3.56	442.01	87.70	±2.86

Table 5. AGC: Arbitrary Data

Actual Class	Assigned Class				P_{cc} (%)	97.5 % Confidence Interval (%)
	9a	aa	f2	fa		
9a	384.54	54.01	1.35	64.10	76.30	±2.02
aa	7.28	490.20	0.00	6.52	97.26	±0.78
f2	10.70	27.82	440.14	36.34	85.46	±1.68
fa	8.62	41.39	2.55	451.43	89.57	±1.45

Table 6. AGC: Train Pristine/Test Arbitrary

Actual Class	Assigned Class				P_{cc} (%)	97.5 % Confidence Interval (%)
	9a	aa	f2	fa		
9a	370.76	74.68	1.68	69.87	71.71	±3.95
aa	1.75	498.74	0.40	5.51	98.57	±1.04
f2	43.63	14.41	541.61	137.36	73.49	±3.87
fa	8.73	34.04	3.41	464.83	90.96	±2.51

4.3 *Impact of Classifier Type*

The results in this section show that some advantage may be gained by using non-parametric techniques, although the improvement is not particularly impressive. Results are given in Table 7 for both the arbitrary and pristine cases. Note that once the alignment problem is "solved," results for the neural network technique converges, while the Gaussian classifier and k-NN still show the influence of the bad data points. Because just fifty repetitions are accomplished through the cross-validation process, the 97.5% confidence interval is on the order of ± 5 % and results shown do not have statistical separation. It seems that the Gaussian classifier is adequately modelling the data. There is a difference in values, however.

Table 7. Comparison of Classifiers

Classifier	Pristine Case ($P_{cc}(\%)$)	Arbitrary Case ($P_{cc}(\%)$)
k-NN	96.48	94.85
Neural Network	96.57	96.55
Gaussian	95.75	92.74

4.4 *Impact of Data*

In this section, results from the Bayes bounding experiments (Table 8) and 2-class AGC runs (Table 9) are presented and compared. Dashes indicate that the k-NN "learned" the data. In other words, it perfectly reclassified the data for all cross validation runs and all values of k. The confidence intervals on Bayes estimates are given by Martin to be 2.26 % for a 97.5 % level of confidence.

Table 8. Bayes Bound Estimates (Option 2): Class by Class

Classes	Pristine Case $P_{cc}(\%)$	Hand Aligned Case $P_{cc}(\%)$	Arbitrary Case $P_{cc}(\%)$
9a—aa	99.55	99.10	98.50
9a—f2	–	99.72	99.65
9a—fa	95.25	94.20	94.50
aa—f2	–	99.80	99.99
aa—fa	–	99.05	99.10
f2—fa	99.94	–	–

Table 9. AGC: Class by Class

Classes	Pristine Case		Hand Aligned Case		Arbitrary Case	
	$P_{cc}(\%)$	Conf(%)	$P_{cc}(\%)$	Conf(%)	$P_{cc}(\%)$	Conf(%)
9a—aa	94.01	±2.06	93.23	±2.11	92.93	±2.25
9a—f2	98.32	±1.13	97.42	±1.38	94.31	±2.03
9a—fa	90.85	±2.22	88.81	±2.76	91.30	±2.47
aa—f2	97.96	±1.16	98.12	±1.18	95.56	±1.81
aa—fa	96.44	±1.62	94.74	±1.96	94.78	±1.95
f2—fa	97.98	±1.23	97.09	±1.46	94.10	±2.06

Several observations may be made from these results. First, the data in a class to class sense appears to be nicely separable. Only the error rates of the **9a – fa** combination seems to overlap in the two class feature space. Also note **f2** seems to show adequate separation in each two class situation. The high performance seen here is made in light of the fact that the alignment problem is "solved," as described in the last chapter.

One may compare Table 9 with the tables from Section 4.2 to see how performance jumps in this simpler, two-class problem. But even this result is affected by the alignment issue, even though alignment plays its normal role in the results of Table 9. In the two class situations, there is only one class to be improperly aligned against. Note how class **f2** is not confused with other classes to the same extent as in the four class case, although it is still the second worst performer. Class **fa** still performs relatively poorly. Also note that alignment is the issue that is intertwined in this specific case because it is the class that the circular centroid seems to struggle with the most.

4.5 Feature Discrimination

Feature discrimination using the EDBFM was performed in the transformed, "EDBFM" space, as well as the original space.

4.5.1 Discrimination in the Transformed Space. The results were generated by utilizing subsets of eigenvectors sorted by magnitude of eigenvalue. These subsets form the

columns of the transformation matrix and are applied to the data as described in the last chapter.

Figure 12 shows results for a two class problem and four class problem. This diagram shows excellent classification rates using just five to ten percent of the available features in the transformed space. The peaking effect seen in the four class case is troubling because all that has been done is a linear transformation. An explanation for the roll off may be that the the transformation is inadequately represented because of the assumption of uncorrelated featres to begin with. In other words, the data is inadequately represented by that assumption and more data is required to properly calculate the full covariance matrix. In the end, it appears that the new feature space is vulnerable to the curse of dimensionality issue discussed in Chapter 2.

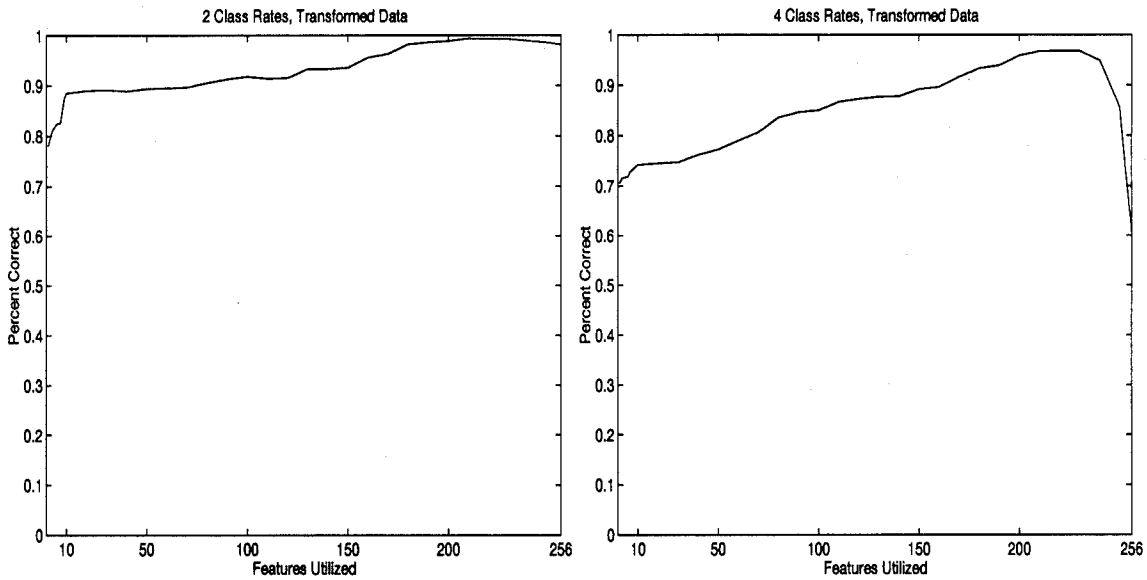


Figure 12. Reclassification Curves, Transformed Data

4.5.2 Discrimination in the Original Space. This section demonstrates the capability to reclassify in the original space by interpreting the most significant eigenvectors of the EDBFM. The average of the ten most significant eigenvectors is shown in Figure 13. As described in Chapter 3, the elements of the eigenvector selected first are those with the highest

magnitude. These elements indicate the features to focus on during reclassification. In the two class problem, it appears that different areas along the signature provide important information for discrimination. One would have to know the specifics of the aircraft under test and the test environment to make in-depth inferences. In both cases the eigenvectors represent an average representation of the significant features with respect to each two class situation. Thus, it is hard to interpret specific areas of interest. However, it seems clear that in the four class case, leading edge information tends to dominate the decision. In the two class case, there are many different areas that provide discriminantly relevant information. Note that the eigenvectors emphasize very different sets of features in each problem. This is not surprising because the calculation of the EDBFM is very data dependent.

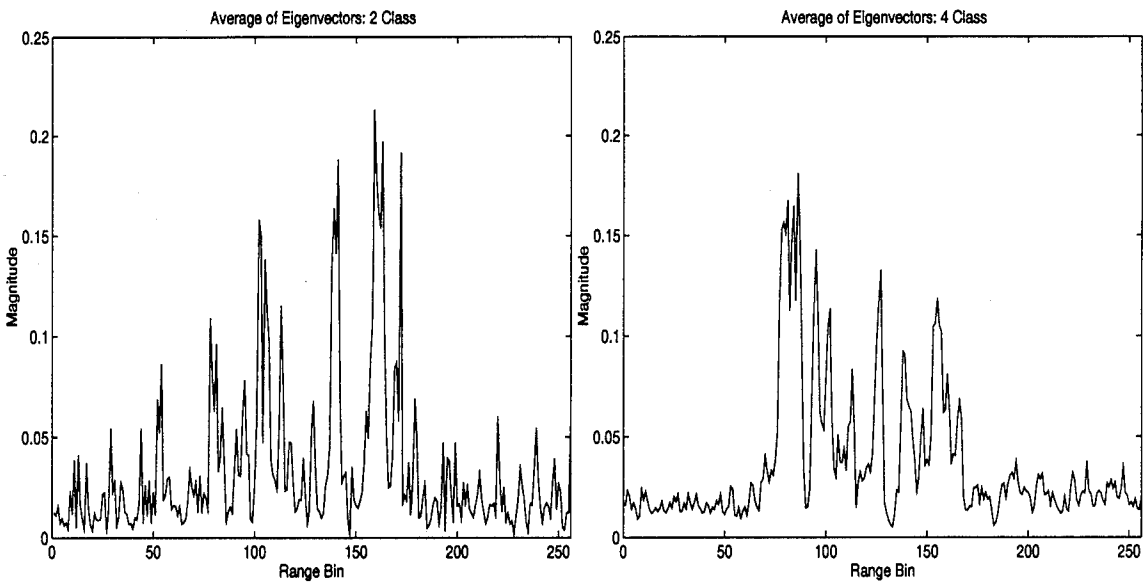


Figure 13. Average of Ten Most Significant Eigenvectors

Figure 14 shows the resulting reclassification rates. Once again, results are quite good. In fact, when using less than half the features, reclassifying in the original space out performs reclassifying in the transformed space. Reclassifying in the transformed space tends to peak at a higher level than reclassifying in the original space, however. Once again, in the four class environment there is a peaking and roll off of performance as more and more features are

added. In this case, features are treated as uncorrelated (via the main-diagonal-only covariance matrix), so the roll off is not as pronounced. Recall from Chapter 2 that this effect is predicted by Chandrasekaran (5) and has to do with the curse of dimensionality.

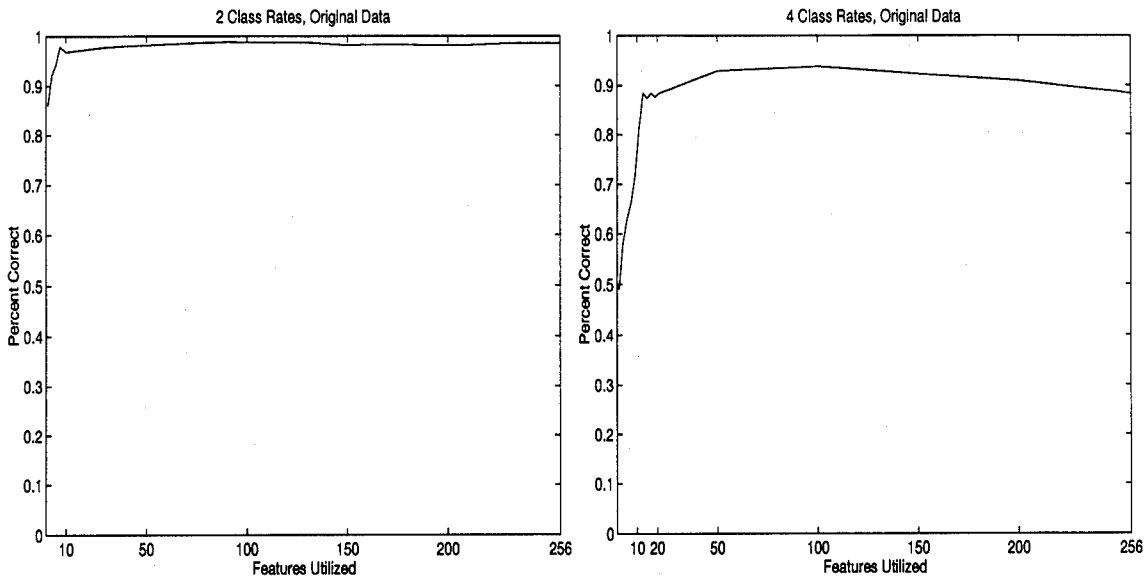


Figure 14. Reclassification Curve, 4-Class, Pristine Data: Original Space

4.6 Summary

This chapter shows all the results which accomplish the most of the goals of thesis: to form an estimate of the AGC's performance with respect to alignment and to perform feature saliency analyses. Results show that alignment is the critical issue in the UHRR radar problem. Classification rates drop over 11% for the **f2** class. In the overall classification rate, though, degradation is on the order of 3%.

Comparison of parametric and non-parametric classifiers shows that there is some, but not a major, price to be paid with the assumption of Gaussian distributions. Classification rates hover around 95% and are not statistically significant at a 97.5% level of confidence.

Despite data that is well separated (as indicated by classification rates above 99% during the Bayes bounding experiments), the ability to properly declare a starting point for the target

within a radar signature is critical to the analysis. The approach used by the AGC intertwines the alignment judgment with the final classification judgment when the issues may be more properly handled separately. The starting point of a target in a noisy signal should be unrelated to its actual class. Thus, the decision of where the signal begins and what the signal is should also be distinct.

Furthermore, this chapter implements and applies Lee's analysis of the EDBFM to the UHRR radar problem. His analysis is extended in this thesis by interpreting the eigenvectors of the EDBFM to identify relevant features in the original feature space. True feature reduction in the original feature space is attained with results meeting or exceeding performance with the full set of features. This can be attained using as little as 5% of the original features (see Figure 14).

In this problem, classification in the original space by interpreting the eigenvectors of the EDBFM outperforms Lee's original method for small feature sets. As more features are added, the Lee transformation method peaks above the method presented in this thesis. However, because features become correlated, performance is hurt as the number of features rises past 225. Both methods achieve peak performance which meet or exceed classification in the original space with a full set of features. Classification of data in the original space shows excellent results here and has been used successfully with other UHRR class combinations and mammography data (25). Chapter 5 extends the use of Lee's algorithm to selecting appropriate wavelet scales with respect to a two class UHRR radar problem.

V. Utilizing Feature Discrimination and Wavelet Transformations

5.1 Introduction

The purpose of this chapter is to use a multiresolution analysis (MRA) of UHRR data and demonstrate that relevant scales may be determined from the EDBFM. Choosing wavelet bands based on discrimination power is shown to have an advantage in classification over choosing wavelet bands with respect to reconstruction error for this data.

5.2 Implementation

5.2.1 Data. The data demonstrating the results are identical to the 2 class problem in Chapter 4, Section 4.5: pristine data, classes **f2** and **fa**. 256-element, raw feature vectors are used because the wavelet code used works best with signals which are powers of two.

5.2.2 Alignment Issues. As mentioned in Chapter 2, implementing a wavelet-based extraction and classification scheme requires special attention to properly aligning data before decomposing them. Classification of UHRR signatures failed when the following process excluded the alignment steps. In this case, the circular centroid is applied only to the original signal to solve the wrap-around problem. Coarse alignment is irrelevant because full correlations are used to align the wavelet features. The overall scheme is shown in Figure 15. To accomplish this, wavelet extraction was performed in a two step process.

First, Suzuki's algorithm is applied to form shift-invariant wavelet representations. Daubechies 20-tap wavelets are used. The problem of edge effects is handled by extending the signal on each end by the average of the nearest 50 bins. In other words, instead of zero padding or periodizing, an average of the first 50 range bins is taken and concatenated to the "front" of the signal. Likewise, an average of the last 50 bins is taken and concatenated to the "tail" of the signal. The averaging was found necessary because the signatures are not nearly equal at their ends and periodic extensions of the signal would introduce discontinuities into

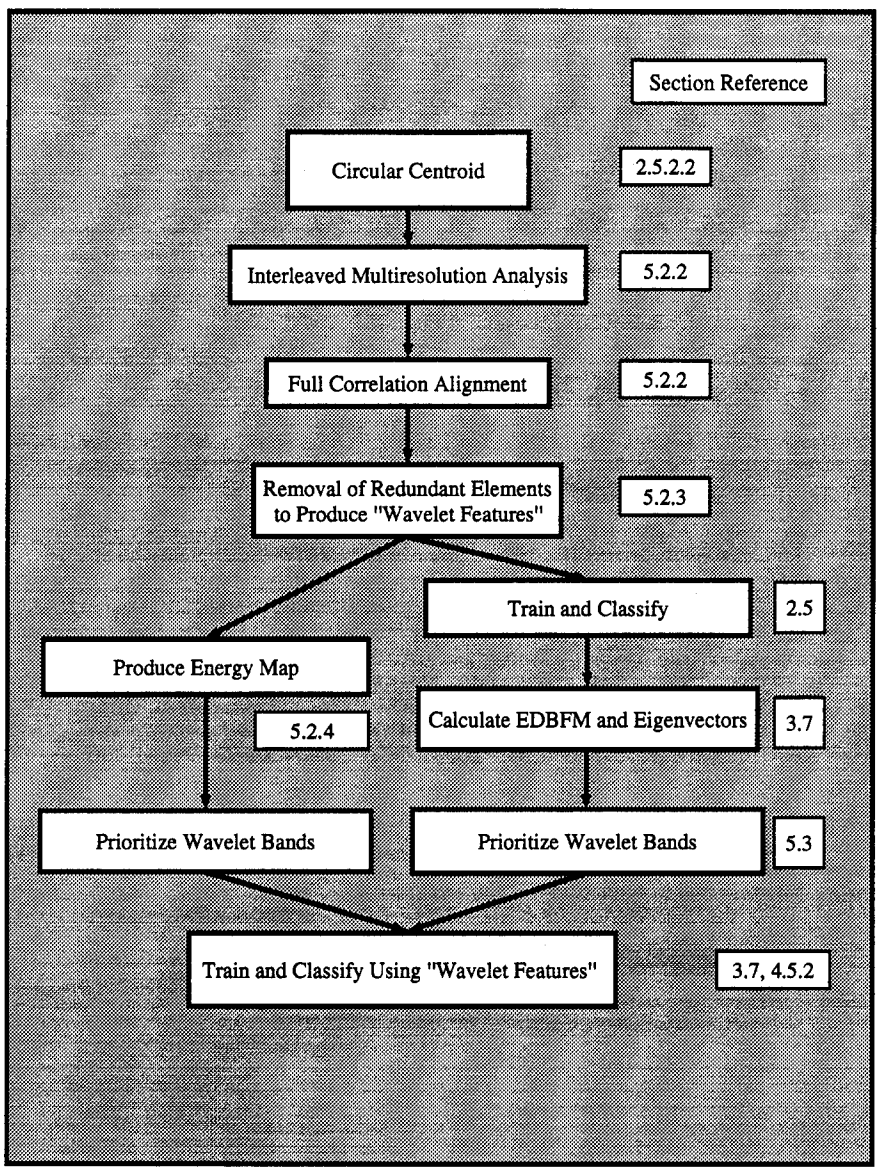


Figure 15. Wavelet Feature Extraction

the analysis. Also, signal extension, as opposed to periodization, is more compatible with Suzuki's code.

Equal numbers of points are added to the beginning and end of the signal to form a signal of length 1024. Suzuki's code assumes zero padding beyond this range. Thus, there will be more pronounced edge effects where the elongated signal goes to zero. The linear convolution process causes edge effects to creep slowly toward the center of the elongated signal with successive filtering. But these edge effects are removed by truncating the signal back to length 256.

Each vector is fully decomposed into eight scales. These resulting vectors are all 256 elements long and contain redundant information about the signal. However, they also include information that can be used to align exemplars. During the correlation process, each level of decomposition is circularly correlated with the corresponding level of decomposition in the class template. The best match is stored in a feature matrix which ends up being 16 by 256 elements holding 8 approximation levels and 8 detail levels.

5.2.3 Formation of Wavelet Feature Vectors. Each level of decomposition in this wavelet feature matrix is then downsampled to remove terms containing redundant information. The first level of decomposition is downsampled by 2, the second level by 4, etc. This is done because Suzuki's code, essentially, has interleaved the wavelet decompositions together (46). All of the downsampled detail coefficients and the eighth level approximation coefficient are retained. The results from this downsampling process are stored in a final, 256 element feature vector. This vector contains all of the downsampled detail signals and the single, lowest level, approximation coefficient. Figure 16 shows the organization of a wavelet feature vector.

These wavelet features are used for training and testing in a Gaussian classifier, as in the rest of the thesis. During training, the wavelet exemplar is added to a recursively calculated template. During testing, a 256-element wavelet exemplar is aligned against each class template before declaring an assigned class. The Lee analysis is applied using the template

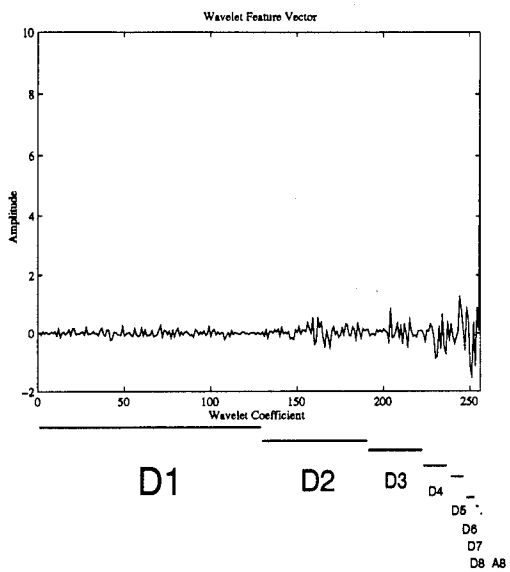


Figure 16. Correspondence of Feature Vector to Wavelet Scales

(containing mean and variance information for each element of the wavelet feature vector) as estimates for the Gaussian classifier. Correctly classified vectors are used to generate the EDBFM and its eigenvectors.

5.2.4 Examination of the Energy Map. To extract the full energy map described by Chang, the wavelet packet recursion is applied on wavelet feature vectors from the training set. For this analysis, code by Myers (35) is used, again with Daubechies 20 tap filters. As mentioned in Chapter 2, this is a recursive filtering process applied to all approximations and all details. In this case, edge effects are handled by mirror-extending the signal being decomposed and performing circular convolutions. The l_1 norm of each leaf in the tree is stored in the energy map and averaged to form the template.

5.3 Evaluation of Significant Wavelet Bands

Wavelet bands are rank ordered separately using the energy map information and the EDBFM information. Classification results using coefficients prioritized by each method are generated.

According to Chang, significant wavelet bands at a given level of decomposition in the energy map are those with highest calculated energy. If this energy surpasses a certain threshold, as a percentage of the highest energy band at that level, then that detail is relevant to the reconstruction of the signal. The idea is applied in this thesis by noting that at each scale, the approximation coefficients had the most significant l_1 norm. This is not surprising because that is where the DC information is. Also, at each level, the detail coefficients associated with the conventional wavelet decomposition (those details not derived as a part of the wavelet packet recursion), had the second most significant bands. The energy of the i^{th} approximation or detail level is designated by E_{ai} or E_{di}, respectively. E_{di} is compared to E_{ai} and relative percentages are recorded to rank order the significance of the wavelet coefficients.

Additionally, energy values are compared across scales by normalizing the l_1 measure by a factor of $\sqrt{2}$ for each additional level of decomposition. This is done because Myers' code does not include the inverse of that term in his filtering routine. As shown in Table 10, the ranking methods agree. Note that the single approximation coefficient is actually ranked more importantly than all the detail levels with an energy measure of .7433.

Table 10. Wavelet Selection: Entropy

Detail Level	E_{di}/E_{ai}	Rank Order	Normalized l_1 Energy	Rank Order
1	.0782	7	.0828	7
2	.1100	3	.1165	3
3	.0964	5	.1022	5
4	.0999	4	.1057	4
5	.0798	6	.0843	6
6	.0693	8	.0719	8
7	.1700	2	.1498	2
8	.2500	1	.1857	1

Alternatively, discriminantly relevant features according to the EDBFM are predicted. The EDBFM's eigenvectors will be interpreted to highlight those wavelet coefficients that are discriminantly relevant. Since the interpretation of eigenvectors points to individual features, instead of groups of features, some measure must be applied to select relevant scales. The

criterion is to take the average energies of the portions of the eigenvectors that correspond to each wavelet band. The most significant eigenvector is used for interpretation. The correspondence of its elements to the wavelet bands is shown in Figure 17. The prioritization results are shown in Table 11 and are compared with the results from the entropy-based analysis. The methods choose different rankings for the scales.

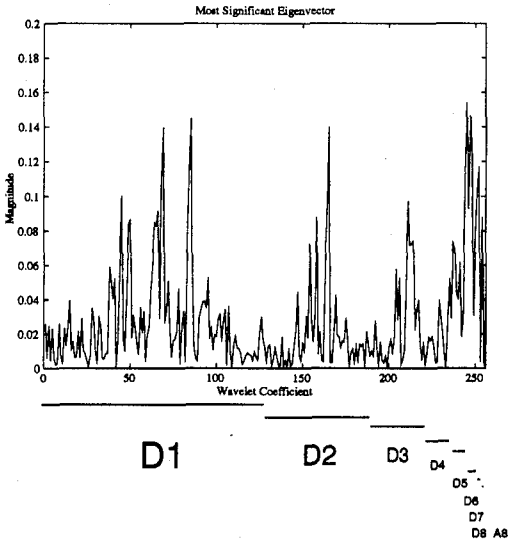


Figure 17. Correspondence of Eigenvector to Wavelet Bands

Table 11. Wavelet Selection: EDBFM Eigenvector Interpretation

Detail Level	EDBFM Eigenvector Analysis	EDBFM Rank Order	Entropy Rank Order
1	.0409	5	7
2	.0284	7	3
3	.0383	6	5
4	.0450	4	4
5	.1396	1	6
6	.1231	2	8
7	.0694	3	2
8	.0043	8	1

5.4 *Classification Using Specified Wavelet Scales*

The test sets containing wavelet features are now submitted for reclassification as in the case of reclassification in the original space of Chapter 4. Results are generated by utilizing only those scales indicated by Table 11 and are shown in Tables 12 and 13. In the tables, "Dx" means the detail level x and "Ax" means the approximation level x.

Table 12. Wavelet Classification Results: Entropy

Levels Included	Number of Features	Classification Accuracy (%)
A8,D8	2	70.91
A8,D8,D7	4	91.44
A8,D8,D7,D2	68	92.57
A8,D8,D7,D2,D4	84	92.82
A8,D8,D7,D2,D4,D3	116	93.07
A8,D8,D7,D2,D4,D3,D1	244	92.44
A8,D8,D7,D2,D4,D3,D1,D5	252	92.56
A8,D8,D7,D2,D4,D3,D1,D5,D6	256	92.56

Table 13. Wavelet Classification Results: EDBFM Eigenvector Interpretation

Levels Included	Number of Features	Classification Accuracy (%)
D5	8	94.33
D5,D6	12	94.08
D5,D6,D7	14	93.95
D5,D6,D7,D4	30	94.45
D5,D6,D7,D4,D1	158	93.20
D5,D6,D7,D4,D1,D3	190	92.56
D5,D6,D7,D4,D1,D3,D2	254	92.56
D5,D6,D7,D4,D1,D3,D2,D8,A8	256	92.56

The EDBFM shows maximum performance with four detail scales and 30 feature elements. In fact, using just one scale, D5, one can out perform the classification using the entire feature set. The results given in the tables show that definite advantage is gained by transforming data specifically with respect to classification error.

5.5 *Summary*

This chapter has shown that relevant wavelet scales with respect to classification may be chosen using decision boundary analysis. These results are compared against Chang's method which uses an entropy measure to determine relevant wavelet scales. The entropy measure technique has recently been updated by Coifman to be geared towards classification error instead of reconstruction error (9). Even though both ranking methods used here produce classification rates that peak above 90%, these results show the disparity between feature discrimination techniques. Also, the applicability of the interpretation of EDBFM eigenvectors to feature reduction in the original (non-transformed) space is verified. Eigenvector interpretation has selected one wavelet scale containing eight feature elements that produces 94% accuracy. This accuracy is comparable to, but not as good as, the result obtained in Chapter IV, where original data produced a classification rate near 97% with only eight features.

VI. Conclusion

6.1 Introduction

For UHRR radar, the underlying problem of aligning exemplars to templates for training and classification is a formidable challenge. In the face of this challenge, this research had a two-fold purpose. The initial portion of the thesis examined the AGC with respect to properly aligning exemplars to class templates. A baseline of the AGC's performance was sought to assess the impact of the alignment problem. The second line of exploration looked at feature discrimination in the context of the UHRR radar problem. This included applying alternative feature selection techniques and the use of an MRA to find discriminantly relevant features and thus, scales.

The techniques used to accomplish these goals come from several broad theoretical areas. The first goal relies on the fundamental theories from statistical pattern recognition. The second goal relies on insights from linear algebra and uses wavelets in an MRA of the data. The fundamentals and literature search associated are presented in Chapter II. Chapter II provides theory for each of the methods applied in Chapter III and the MRA used in Chapter V. Chapter IV presents the results for AGC baselining, including alignment analysis, classifier analysis and data analysis. Also, feature selection with respect to the decision boundary is developed. The selection algorithm is extended by suggesting a method for using features in the original space that does not require linear combinations of the features. The new method selects specific features in the original space that are discriminantly relevant. Chapter V applies similar feature selection analysis to choosing relevant wavelet scales.

6.2 Summary of Key Results

6.2.1 AGC Baseline. An analysis of the AGC training and testing algorithms shows how UHRR radar signatures are processed and classified. It is shown that final alignment and classification occur at the same time in the AGC. Thus, essentially, the same information

is being used to make both assessments about a data signature. This analysis appears in Section 2.5.

The four class UHRR radar problem is rigorously analyzed. The results show that alignment has a statistically significant impact on the AGC's ability to classify data (Section 4.2). Because alignment is directly associated with the statistical decision for class membership, the alignment issue will play a role in any classifier applied in the same way. Independent of alignment, excellent classification rates above 95% are produced with Gaussian and non-parametric approaches (Section 4.3). The data itself is shown to be reasonably separable, independent of the alignment issue, because the Bayes bounding analysis gives classification rates in the 98^{th} percentile for all two class combinations (Section 4.4).

6.2.2 Feature Discrimination. The feature selection technique given by Lee and Landgrebe is applied successfully to two UHRR radar situations and results achieve classification rates on the order of 90% for substantially reduced feature combinations (Section 4.5.1). However, the roll-off in the the four class situation highlights that the new feature set is a linear combination of the original features. This roll-off also demonstrates that redundant features can detract from a classifier's performance. The linear dependence problem is solved and a technique is demonstrated which truly reduces the number of features required for classification with rates that meet or exceed classification with a full feature set (Section 4.5.2). It is shown that as fewer than 5% of the features in the *original* feature space may be used to attain classification rates of over 95% in the two class case, and nearly 90% in the four class case . This new technique interprets the orientation of the eigenvectors to deduce which feature elements are most relevant to the classification problem (Section 3.7).

6.2.3 Selection of Relevant Scales in an MRA. The new feature selection technique is applied to an MRA and successfully selects the discriminantly relevant scales to produce classification rates on the order of those attained using the raw UHRR radar data. For limited feature sets, it is shown that its performance exceeds that of an entropy based measure of feature significance.

6.3 *Conclusions*

The results of this thesis imply the following:

1. Alignment plays a critical role in the approach of the AGC towards the UHRR radar problem. The AGC approach inextricably intertwines the alignment process and the classification decision making process. Independent of alignment, UHRR radar data is separable on the order of 95% classification rates, with or without the assumption of Gaussian distributions. The starting point of a signal should be found independent of its class to improve performance of the AGC.
2. For UHRR radar data, certain portions of an aircraft are more discriminantly relevant than others. Decision boundary analysis reveals relevant features can vary from problem to problem. The eigenvector interpretation technique can be used to highlight salient airplane characteristics between classes. Further research applying the techniques in this thesis to various combinations of classes and data sets is required.
3. Wavelet analysis shows promise for discriminating UHRR radar signatures based on scale information. Classification rates above 90% are comparable to classification in which the raw data is used. Also, scale analysis is directly related to the properties of the target. Extension of the feature selection technique developed in this thesis to full wavelet packet bases offers great potential. Shift invariant wavelet bases may be especially suitable to the specific UHRR radar problem.

These conclusions show that this thesis meets the objectives laid out in Section 1.5. Exciting implications for feature selection based on the orientation of the decision boundary are applied and hold promise for a wide variety of pattern recognition problems. The problem analyzed here is made more difficult by its high dimensionality and alignment issues. Application to other, more refined problems will give more insights into the EDBFM techniques presented and may yield a generalizable theory for reducing the dimensionality of feature spaces.

Appendix A. Demonstration of Discrimination with the EDBFM

This appendix provides two sample problems which show the implementation of Lee's algorithm with respect to feature discrimination. Each of these cases is a two class problem. The vectors given as bases for the transformed, EDBFM space, are off by 180 degrees from Lee's results, but this is due only to the relative directional comparisons between classes. The Matlab code producing the randomized data and results is given in Appendix C.

A.1 Sample Problem 1: 2 Dimensions

The data in this example have the following statistics, where the means are represented by M_i and covariances are represented by Σ_i.

$$M_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\Sigma_1 = \Sigma_2 = \begin{bmatrix} 1.00 & 0.50 \\ 0.50 & 1.00 \end{bmatrix}$$

This data is distributed as two ellipses and are shown in Figure 18. The EDBFM is calculated by taking the outer product of normal vectors at discrete points along the decision boundary.

Normal vectors are found by finding the intersection of the directed vector connecting two data points and the equation for the decision boundary. One cycles through each data point in each class, using only corrected classified samples. The first two iterations of this process are shown in Figure 19. The number of circles in the diagram are reduced by the chi-squared test. The dashed line represents the equation for the decision boundary. Again, as indicated in the text of the thesis, the outer products of all the normal vectors are averaged to produce the EDBFM. Eigenvalues and eigenvectors of the EDBFM are then taken to yield

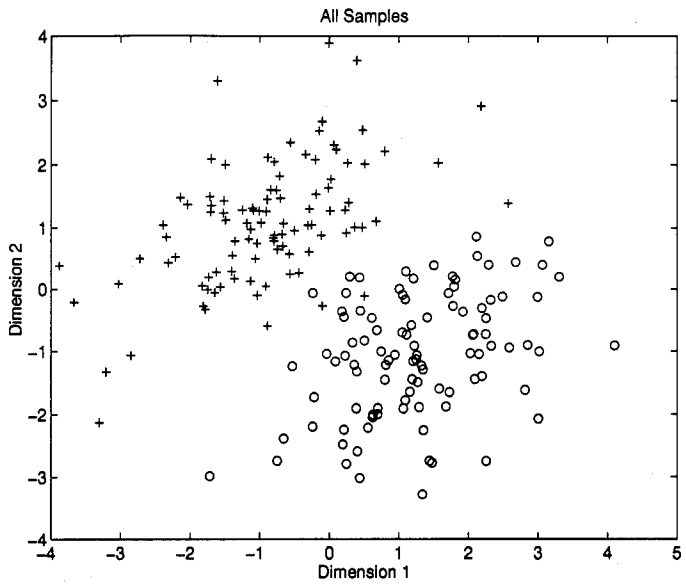


Figure 18. 2 Dimensional Sample Problem

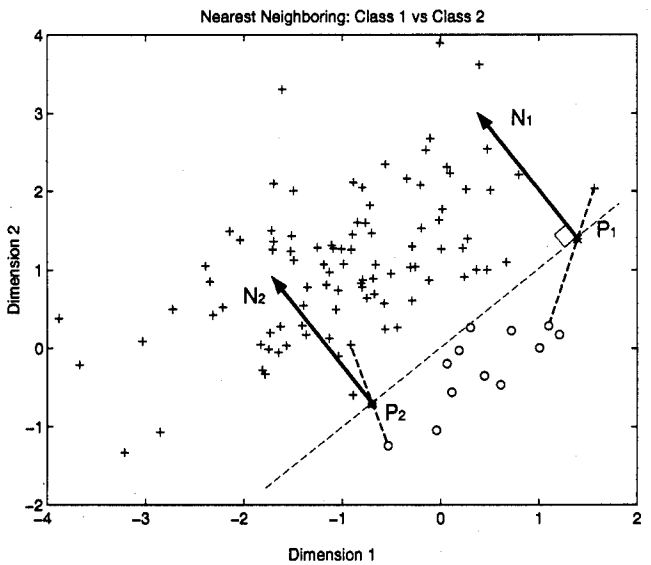


Figure 19. Calculation of EDBFM

information for the transformation and feature reduction. In this case the results are:

$$\text{EDBFM} = \begin{bmatrix} .25 & -.25 \\ -.25 & .25 \end{bmatrix}$$

$$\Phi = \begin{bmatrix} -.7071 & -.7071 \\ .7071 & -.7071 \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} .5000 & .0000 \\ .0000 & -.0000 \end{bmatrix}$$

where Φ holds the eigenvectors in its columns and Λ holds the associated eigenvalues along its main diagonal. In Figure 20, one may see the directions of relevance by visualizing the eigenvectors given in the Φ matrix onto the graph. Figure 20 shows how the decision boundary is ultimately estimated by connecting data points and finding the intersection with the boundary equation.

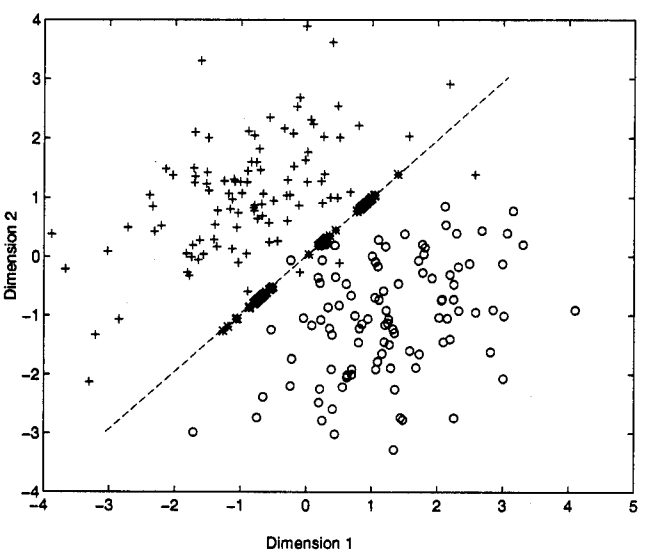


Figure 20. Estimated Decision Boundary

A.2 *Sample Problem 2: 3 Dimensions*

This section shows the algorithm work in a three dimensional situation. Plots show the xy-plane only, so data has been projected onto the paper. Note that this is a convenient analogy with what happens during the EDBFM transformations. The third dimension is coming out of the page, but is irrelevent to classification.

The statisitics for the data are as follows:

$$M_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 3.00 & 0.00 & 0.00 \\ 0.00 & 3.00 & 0.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

$$\Sigma_2 = \begin{bmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

In two dimensions, the data looks as shown in Figure 21. The other two figures show information in the same way as the previous sample problem.

The resulting computations yield:

$$\text{EDBFM} = \begin{bmatrix} .2272 & -.0323 & .0000 \\ -.0323 & .2728 & .0000 \\ .0000 & .0000 & .0000 \end{bmatrix}$$

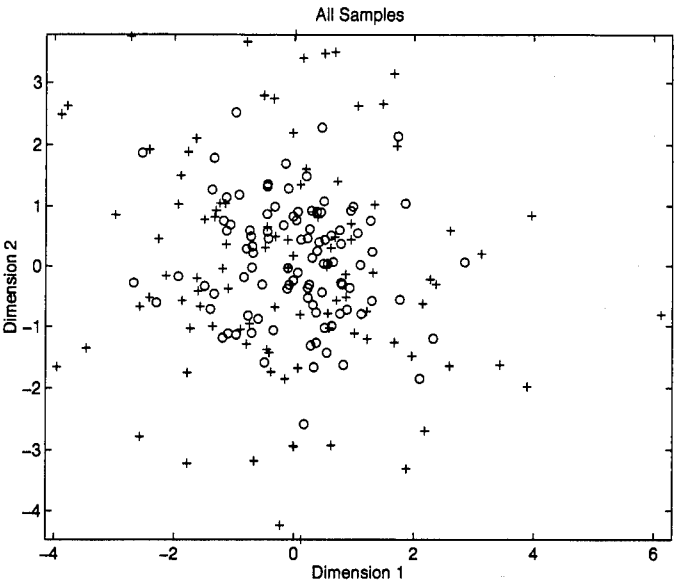


Figure 21. 3 Dimensional Sample Problem (xy Plane)

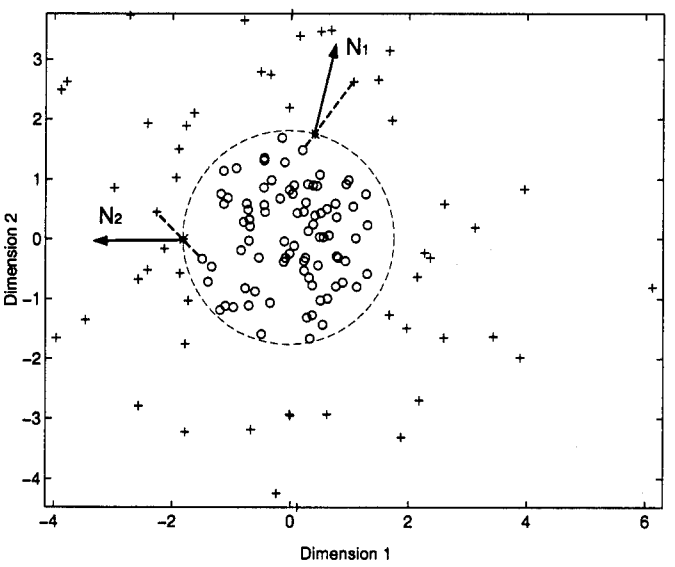


Figure 22. Calculation of EDBFM

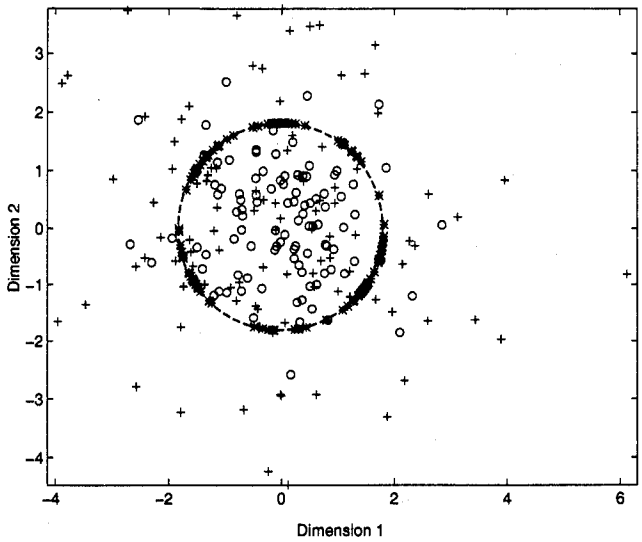


Figure 23. Estimated Decision Boundary (xy Plane)

$$\Phi = \begin{bmatrix} -.7978 & -.6030 & .0000 \\ .6030 & -.7978 & .0000 \\ .0000 & .0000 & 1.0000 \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} .2854 & .0000 & .0000 \\ .0000 & .2146 & .0000 \\ .0000 & .0000 & .0000 \end{bmatrix}$$

In this case it is easy to see that two of the dimensions are required for discrimination in the original space or in the transformed space. This sample problem shows that higher amplitude elements of the significant eigenvectors "point to" the key dimensions in the original space, i.e. the first two elements are "high" relative to the third element.

Appendix B. Alternative Training Methods for the AGC

This appendix provides two alternative training methods which show how classification rates may be improved.

B.1 Recursive Coarse Alignment of Template

B.1.1 Introduction. In the calculation of a class template during training, exemplars are finely aligned to the cumulative template and then added into the template. This fine alignment shifts the current exemplar away from its true power centroid, as calculated by the coarse alignment routine. When this off-centroided signature is included in the template, the resultant class template no longer has a power centroid located at the center bin. At the end of the training process, usually involving five hundred or more training signatures, the resultant power centroid of the mean template was found to be 8 to 18 bins away from its center bin.

It was conjectured that this offset is enough to prevent some in-class samples from being properly aligned because the centroid of the mean was no longer centered. This "centroid drift" problem was corrected by recursively circularly centroiding the mean template after each current exemplar was added in. The C code of the AGC was modified to accomplish this goal.

B.1.2 Results. Results are presented in the following tables for the three cases indicated. Results were generated by using the same methodology given in Chapter 3. Comparisons with the other AGC runs in Chapter 4 are made in the last section of this appendix.

B.2 Full Correlations

B.2.1 Introduction. As was described in Chapter 2, correlations are restricted to shifts of ± 18 bins during training. (During testing, this limited shift range helped prevent

Table 14. AGC: Pristine Data, Recursive Alignment

Actual Class	Assigned Class				P_{cc} (%)	95 % Confidence Interval
	9a	aa	f2	fa		
9a	379.80	50.03	1.46	61.70	77.03	±1.91
aa	1.55	500.10	0.00	2.34	99.22	±0.39
f2	0.55	11.35	277.87	5.21	94.19	±1.10
fa	8.11	30.91	3.72	456.24	91.43	±1.67

Table 15. AGC: Arbitrary Data, Recursive Alignment

Actual Class	Assigned Class				P_{cc} (%)	95 % Confidence Interval
	9a	aa	f2	fa		
9a	383.47	55.93	1.15	63.43	76.08	±2.15
aa	2.22	498.50	0.00	3.27	98.91	±0.53
f2	1.31	21.16	471.62	20.90	91.57	±1.40
fa	7.99	31.21	2.34	462.47	91.76	±1.39

Table 16. AGC: Train Pristine, Test Arbitrary, Recursive Alignment

Actual Class	Assigned Class				P_{cc} (%)	95 % Confidence Interval
	9a	aa	f2	fa		
9a	379.90	74.01	1.44	61.64	73.48	±2.05
aa	2.60	500.07	0.00	3.33	98.82	±0.49
f2	9.41	19.87	641.49	66.22	87.04	±1.56
fa	8.24	34.92	4.42	463.43	90.68	±1.54

interclass confusion). This experiment allows for full correlations during fine alignment in the training process.

B.2.2 Results. Results are presented in the following tables for the three cases indicated. Results were generated by using the same methodology given in Chapter 3. Comparisons with the other AGC runs in Chapter 4 are made in the last section of this appendix.

Table 17. AGC: Pristine Data, Fully Correlated Alignment

Actual Class	Assigned Class 9a	aa	f2	fa	P_{cc} (%)	95 % Confidence Interval
9a	380.52	50.21	0.87	61.40	77.19	±1.92
aa	1.57	500.11	0.00	2.32	99.22	±0.40
f2	.818	11.47	277.05	5.65	93.91	±1.09
fa	8.29	31.27	2.62	456.81	91.55	±1.37

Table 18. AGC: Arbitrary Data, Fully Correlated Alignment

Actual Class	Assigned Class 9a	aa	f2	fa	P_{cc} (%)	95 % Confidence Interval
9a	386.70	53.658	0.55	63.58	76.73	±2.11
aa	2.16	498.63	0.00	3.20	98.93	±0.51
f2	1.49	21.41	466.89	25.20	90.66	±1.45
fa	8.07	31.26	1.61	463.06	91.87	±1.52

B.3 Leave One Out Results

Leave one out results were generated for a single ordering of the data. That is, the order of presentation to the training algorithm remained the same throughout. These results are consistent with the results attained in the overall runs. The original AGC technique (without the recursive alignment or full correlation options) was used in the pristine and arbitrary cases. The pristine case resulted in a 92.29% classification rate and the arbitrary case resulted in a 89.45% classification rate. As predicted by Devijver, the hold out runs done earlier in the thesis provide pessimistic estimates of the error rates. They predict higher rates than would be

expected. Part of this is due to the problem of using test sets with some common test vectors over the trials.

B.4 Summary

This section compares results presented in this appendix with the original runs. The "mixed" case in Table 19 refers to training on pristine data and testing on arbitrary data. See Chapter 3 for full definitions and methodology.

Some observations are worth pointing out. First, there is a general improvement in performance with respect to the arbitrary data. In the mixed case, there is a definite jump in the classification rate, showing that the previous inability to learn the alignment problems is corrected. Third, relative class performance generally remains the same except that the full correlation runs associated with class **f2** show marked improvement.

Table 19. Comparison of Results

Case	P_{cc} (%)	97.5 % Confidence Interval (%)
Pristine (AGC)	90.16	±1.31
Arbitrary (AGC)	87.13	±1.59
Pristine (Recursive)	90.12	±1.35
Arbitrary (Recursive)	89.59	±1.54
Pristine (Full Correlation)	90.14	±1.37
Arbitrary (Full Correlation)	89.56	±1.53
Mixed (AGC)	82.60	±3.32
Mixed (Recursive)	87.40	±1.54

Appendix C. Computer Code

This appendix provides Matlab code which may be used to implement Lee's EDBFM technique in a 2 and 4 class problem.

C.1 Sample Problem Code

This section contains code to produce the sample problems found in Appendix A. It should run without modification. All plot commands have been removed.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program demonstrates the Lee and Landgrebe algorithm,
% as presented in their article.
clear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set Environment Parameters

% Choose number of samples per class
numsigs = 100;

% Choose which sample problem to run
option = 3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set Statistics for each option
if option==1;
dimen=2;
rand('seed',0);
rc1 =randn(numsigs,dimen);
rand('seed',2);
rc2 =randn(numsigs,dimen);

m1 = [-1 1]';
m2 = [1 -1]';
v1 = [1 .5;.5 1];
v2 = [1 .5;.5 1];
end

if option==2;
dimen=2;
rand('seed',0);
rc1 =randn(numsigs,dimen);
rand('seed',2);
rc2 =randn(numsigs,dimen);

m1 =[.05 0]';
m2 = [-.05 0]';
v1 = [3 0;0 3];
v2 = [3 0;0 1];
end

if option==3;
dimen=3;
rand('seed',0);
rc1 =randn(numsigs,dimen);
rand('seed',2);
rc2 =randn(numsigs,dimen);

m1 =[0 0 0]';
m2 = [0 0 0]';

v1 = [3 0 0;0 3 0;0 0 1];
v2 = [1 0 0;0 1 0;0 0 1];
end

if option==4;
dimen=4;
rand('seed',0);
rc1 =randn(numsigs,dimen);
rand('seed',2);
rc2 =randn(numsigs,dimen);

m1 =[-1 1 -1 1]';
m2 = [-1 1 -1 1]';
v1 = [6 0 0 0;0 6 0 0;0 0 1 0;0 0 0 1];
v2 = [1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate information for colorizing the noise
[V1,d1] = eig(v1);
[V2,d2] = eig(v2);

lambda1 = sqrt(d1);
lambda2 = sqrt(d2);

p1 = .5;
p2 = .5;
cov(rc1);
cov(rc2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Colorize the white noise
if dimen==2
rc1color = V1*lambda1*rc1' + ..
 [m1(1)*ones(1,numsigs);m1(2)*ones(1,numsigs)];
rc2color = V2*lambda2*rc2' + ..
```

```
 [m2(1)*ones(1,numsigs);m2(2)*ones(1,numsigs)];
elseif dimen==3
rc1color = V1*lambda1*rc1' + ..
 [m1(1)*ones(1,numsigs);m1(2)*ones(1,numsigs);m1(3)*ones(1,numsigs)];
rc2color = V2*lambda2*rc2' + ..
 [m2(1)*ones(1,numsigs);m2(2)*ones(1,numsigs);m2(3)*ones(1,numsigs)];
elseif dimen==4
rc1color = V1*lambda1*rc1' + ..
 [m1(1)*ones(1,numsigs);m1(2)*ones(1,numsigs);m1(3)*ones(1,numsigs);m1(4)*ones(1,numsigs)];
rc2color = V2*lambda2*rc2' + ..
 [m2(1)*ones(1,numsigs);m2(2)*ones(1,numsigs);m2(3)*ones(1,numsigs);m2(4)*ones(1,numsigs)];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute means and covariances
sig1 = cov(rc1color');
sig2 = cov(rc2color');
sig1 = v1;
sig2 = v2;

detsig1 = det(sig1);
detsig2 = det(sig2);

invsig1 = inv(sig1);
invsig2 = inv(sig2);

M1 = mean(rc1color');
M2 = mean(rc2color');
M1 = m1';
M2 = m2';

c1 = rc1color';
c2 = rc2color';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialize mahalanobis distances
% and classification matrices

md11 = zeros(numsigs,1);
md12 = zeros(numsigs,1);
md21 = zeros(numsigs,1);
md22 = zeros(numsigs,1);

d11 = zeros(numsigs,1);
d12 = zeros(numsigs,1);
d21 = zeros(numsigs,1);
d22 = zeros(numsigs,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Classify the data using a simple gaussian classifier

for i = 1:numsigs
md11(i) = (c1(i,:)-M1)*invsig1*(c1(i,:)-M1)';
md12(i) = (c2(i,:)-M1)*invsig1*(c2(i,:)-M1)';
md21(i) = (c1(i,:)-M2)*invsig2*(c1(i,:)-M2)';
md22(i) = (c2(i,:)-M2)*invsig2*(c2(i,:)-M2)';

d11(i) = .5*md11(i) + .5*log(detsig1);
d12(i) = .5*md12(i) + .5*log(detsig1);
d21(i) = .5*md21(i) + .5*log(detsig2);
d22(i) = .5*md22(i) + .5*log(detsig2);
end

b1 = .5*log(detsig1);
b2 = .5*log(detsig2);

correct1 =0;
correct2 = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Classify the data and sort the data into
% good and bad groups
% The good group is retained for later analysis
wch1 = [];wchbad1=[];
wch2 = [];wchbad2=[];

for j=1:numsigs

if d11(j) < d21(j)
correct1=correct1+1;
wch1 = [wch1; j];
else
wchbad1 = [wchbad1; j];
end

if d22(j) < d12(j)
```

```
correct2=correct2+1;
wch2 = [wch2; j];
else
wchbad2 = [wchbad2; j];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Retain distance and classification information
% for correctly classified samples

md11t = md11(wch1,:);
md12t = md12(wch2,:);
md21t = md21(wch1,:);
md22t = md22(wch2,:);

d11t = d11(wch1,:);
d12t = d12(wch2,:);
d21t = d21(wch1,:);
d22t = d22(wch2,:);

true1=c1(wch1,:);
true2=c2(wch2,:);
bad1=c1(wchbad1,:);
bad2=c2(wchbad2,:);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Perform chi test to eliminate outliers

% The following is taken from
% Arnold O. Allen pp137-138,625
stat = 'performing xisquare test'
n = 2;
zalpha = 1.6449;
chialpha = 4.9915;

if dimen==3
chialpha = 7.815;
end

if dimen==4
chialpha = 9.4877;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Apply in-class chi-test, retaining relevant data points
% Note here one could have just as easily used mdxxx and
% compared it directly to chialpha

stat='in-class chi-square'

rel1 = true1(chitesttoy(d11t,chialpha+b1),:);
rel2 = true2(chitesttoy(d22t,chialpha+b2),:);

% Apply chi test to other classes
stat='interclass chi-square'
Lmin =5;
chialpha2 = chialpha;

rel12 = true2(Ltesttoy(d12t,chialpha2+b1,Lmin),:);
rel21 = true1(Ltesttoy(d21t,chialpha2+b2,Lmin),:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate EDBFM
% Begin by reorganizing data
stat = 'calculate N and EDBFM'
m1=m1';
m2=m2';

% NOTE that here we use ACTUAL (non-estimated) values
% for the pdf parameters
P = [];
P1 = [];
P2 = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Loop on each class
for i = 1:2

eval(['cofi = rel' int2str(i) ';'])
eval(['cmref = m' int2str(i) ';'])
eval(['cvref = v' int2str(i) ';'])

numcofi = size(cofi,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compare to other class
for j=1:2
```

```
if j~=i
EDBFM = zeros(dimen,dimen);

N = zeros(1,numcofi);
eval(['othc = rel' int2str(i) int2str(j) ';'])
eval(['cmother = m' int2str(j) ';'])
eval(['cvother = v' int2str(j) ';'])

for k = 1:numcofi

closest = findnearesttest(cofi(k,:),othc,cvref);
[N,p] = computeNPtest(cofi(k,:),cmref,cvref,closest,cmother,cvother,.5,.5,i);
P = [P p];

if i==1
P1 = [P1 p];
elseif i ==2
P2 = [P2 p];
end

EDBFM = EDBFM + (1/numcofi)*(N*N');

end

eval(['E' num2str(i) num2str(j) ' = EDBFM;'])
end
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now, combine each into the final, averaged EDBFM

EDBFM = zeros(dimen,dimen);

p1 = .5;
p2 = .5;

for i=1:2
for j=1:2
if j~=i
eval(['EDBFM = EDBFM + p1*p2*E' num2str(i) num2str(j) ';'])
end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate eigenvalues and eigenvectors for future use
[V,d] = eig(EDBFM)

rankdbfm = rank(EDBFM)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Chitesttoy subroutine
function[goodonly] = chitest(x,chialpha)

goodonly = [];

for i = 1:size(x,1)

if x(i) < chialpha
goodonly = [goodonly; i];
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Ltesttoy subroutine
function[goodonly] = Ltest(x,chialpha,Lm1)

goodonly = [];

for i = 1:size(x,1)
size(x,1);
if x(i) < chialpha
goodonly = [goodonly; i];
end
end

chk = size(goodonly,1);

if chk < Lm1
[dum wch] = sort(x);
goodonly = wch(1:Lm1);
end%if
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%findnearesttest subroutine
function[winner] = findnearesttest(reference,testsigs,var)

[numsigs,feats] = size(testsigs);
```

```
bestmatch = 0;
dists = zeros(numsigs,1);
var=var';


for i = 1:numsigs
i;
xtrial = testsigs(i,:);
[lx,wx] = size(xtrial);
[lref,wref] = size(reference);

dists(i) = (xtrial-reference)*(inv(var))*(xtrial-reference)';

end

[Y,maxind] = min(dists);

winner = testsigs(maxind,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% computeNPtest subroutine
%  Implement equations found in Lee & Landgrebe article
%  Note:  orient is used to insure that you are the vector
%  connecting the two sample points are pointed the same
%  way when you switch from comparing class 1 to class 2
%  and vice-versa
function[N,P] = computeNPtest(x1,m1,v1,x2,m2,v2,prob1,prob2,orient)

x1;x2;m1;v1;m2;v2;
x1 = x1';x2 = x2';
m1 = m1';m2 = m2';

v0 = x1;
v = x2-x1;

det1 = det(v1);
det2 = det(v2);

inv1 = inv(v1);
inv2 = inv(v2);

c = .5 * (m1'*inv1*m1 - m2'*inv2*m2) + .5*log( det1/det2 );
cprime = .5*v0' * (inv1-inv2) * v0 - (m1'*inv1-m2'*inv2)*v0 + c;

b = v0'*(inv1-inv2)*v - (m1'*inv1-m2'*inv2)*v;
a = .5 * v' * (inv1-inv2) * v;

t = log(prob1/prob2);

if a == 0
u = (t-cprime)/b;
else
u1 = (-b + sqrt(b^2-4*a*(cprime-t)))/(2*a);
u2 = (-b - sqrt(b^2-4*a*(cprime-t)))/(2*a);
if orient==1
u = min([u1 u2]);
else
u = max([u1 u2]);
end
end

p = u*v+v0;

N = (inv1-inv2)*p + (inv2*m1 - inv1*m2);
P=p;

N = N/sqrt(N'*N);
```

C.2 4-Class Code

This section contains code used in the four class problems of Chapter 4. The first section here finds the EDBFM and its eigenvectors and eigenvalues. The second section gives code for reclassification in the transformed space. The third section gives code for reclassification in the original space. Associated subroutine may be found in the final section of this appendix. Note that the programs are especially designed for the UHRR radar problem (due to alignment). It should not be difficult to generalize the concepts to other problems or simplify it to a two class problem.

C.2.1 EDBFM Analysis.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SET NUMBER OF CLASSES AND NUMBER OF FEATURES USED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Class id's correspond to  1 == 9a
% 2 == aa
% 3 == f2
% 4 == fa

feats = 256

classes=4
n = classes;

troubleshoot = 0;

load trialp256

load paramsprs256

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This section uses a subroutine to find the correctly
%  classified sample from the UHRR training and test sets
%  The files w/ tst prefixes contain test data
if 1==0

load tst9a256
load tstaa256
load tstf2256
load tstfa256

sigs = [tst9a256; tstaa256; tstf2256; tstfa256];

%  Single out correctly classified samples only and store
%  The following variables hold correctly classified samples (cxxg)
%  and the respective test statistics (dxxg) as computed by the AGC

stat = 'finding correctly classified data'
[c9ag caag cf2g cfag d9ag daag df2g dfag maxindices] = ..
 findcorrect(sigs,cum_name,cum_dis,4);
save paramsprs256 c9ag caag cf2g cfag d9ag daag df2g dfag maxindices
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  FOR TROUBLESHOOTING, LIMIT THE NUMBER OF SIGNALS CONSIDERED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if troubleshoot == 1
howmany = 100;
c9a = c9ag(1:howmany,:);
caa = caag(1:howmany,:);
cf2 = cf2g(1:howmany,:);
cfa = cfag(1:howmany,:);

d9a = d9ag(1:howmany,:);
daa = daag(1:howmany,:);
df2 = df2g(1:howmany,:);
dfa = dfag(1:howmany,:);
else
c9a = c9ag;
caa = caag;
cf2 = cf2g;
cfa = cfag;

d9a = d9ag;
daa = daag;
df2 = df2g;
dfa = dfag;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  Eliminate Outliers of each class with Chi-Square Test

%  Find biases and subtract out to get true Mahalanobis distance
stat = 'finding biases'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  IMPORTANT:  PRELIMINARY RESULTS INDICATE YOU MUST
%  RETAIN THE BIAS IN THE DISTANCE CALCULATION BECAUSE
%  YOU MAY OTHERWISE ENCOUNTER TWO POINTS NOT SEPARATED
%  BY THE h(X). RECALL THAT POINTS MUST STRADDLE THE
%  THE DECISION BOUNDARY OR NO INTERSECTION CAN BE FOUND!
%  (BIAS IS THE DETERMINANT OF THE CLASS COVARIANCE)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%  prsxxtemp is a variable representing the respective
%  class of pristine data

[b9a m9a v9a] = findbias('prs9atemp');
[baa maa vaa] = findbias('prsaatemp');
[bf2 mf2 vf2] = findbias('prsf2temp');
[bfa mfa vfa] = findbias('prsfatemp');
d9at =zeros(size(d9ag));
daat =zeros(size(daag));
df2t = zeros(size(df2g));
dfat = zeros(size(dfag));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% HERE, RETAIN ORIGINAL CLASSIFICATION DISTANCE:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
d9at = d9a;
daat = daa;
df2t = df2;
dfat = dfa;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
stat = 'performing xisquare test'


%  The following info is taken from
%  Arnold O. Allen, Prob, Stats, and Queuing
%  theory pp 137-138;and p. 625, table 4

n = feats; %, the # of degrees of freedom
zalpha = 1.6449;        %  95% inclusion
%zalpha = 1.96; %  97.5 inclusion
%zalpha = 2.3263 %  99.9 inclusion
%zalpha = -1.6449; %  5% inclusion
%zalpha = -1.96; %  2.5 inclusion

% The test statistic with n > 100:

chialpha = n*(1-2*(9*n)^(-1)+zalpha*sqrt(2*(9*n)^(-1)))^(3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Apply in-class chi-test, retaining relevant data points
stat='in-class chi-square'

rel1 = c9a(chitest(d9at,chialpha+b9a,1),:);
rel2 = caa(chitest(daat,chialpha+baa,2),:);
rel3 = cf2(chitest(df2t,chialpha+bf2,3),:);
rel4 = cfa(chitest(dfat,chialpha+bfa,4),:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Apply "Chi-Square" test to other classes
stat='interclass chi-square'
Lmin = 50;
chialpha2 = chialpha;

rel12 = caa(Ltest(daat,chialpha2+b9a,1,Lmin),:);
rel13 = cf2(Ltest(df2t,chialpha2+b9a,1,Lmin),:);
rel14 = cfa(Ltest(dfat,chialpha2+b9a,1,Lmin),:);

rel21 = c9a(Ltest(d9at,chialpha2+baa,2,Lmin),:);
rel23 = cf2(Ltest(df2t,chialpha2+baa,2,Lmin),:);
rel24 = cfa(Ltest(dfat,chialpha2+baa,2,Lmin),:);

rel31 = c9a(Ltest(d9at,chialpha2+bf2,3,Lmin),:);
rel32 = caa(Ltest(daat,chialpha2+bf2,3,Lmin),:);
rel34 = cfa(Ltest(dfat,chialpha2+bf2,3,Lmin),:);

rel41 = c9a(Ltest(d9at,chialpha2+bfa,4,Lmin),:);
rel42 = caa(Ltest(daat,chialpha2+bfa,4,Lmin),:);
rel43 = cf2(Ltest(df2t,chialpha2+bfa,4,Lmin),:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The xxxxxtemp files hold template information (mean
% and variances) for each class
stat = 'loading mean and variance info'
%load prs9atemp
m1 = m9a;
v1 = v9a;
%load prsaatemp
m2 = maa;
v2 = vaa;
%load prsf2temp
m3 = mf2;
v3 = vf2;
%load prsfatemp
m4 = mfa;
v4 = vfa;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% OK, keep your fingers crossed
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
stat = 'calculate N and EDBFM '

for i = 1:4

eval(['cofi = rel' int2str(i) ';'])
eval(['cmref = m' int2str(i) ';'])
eval(['cvref = v' int2str(i) ';'])

numcofi = size(cofi,1)

for j=1:4
[i j]
if j~=i
EDBFM = zeros(n,n);

eval(['othc = rel' int2str(i) int2str(j) ';'])
eval(['cmother = m' int2str(j) ';'])
eval(['cvother = v' int2str(j) ';'])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The following scaling trick is required to prevent machine
% precision problems. The factor of 100 divides out in the
% computeNP subroutine
detref = det(diag(100*cvref));
detother = det(diag(100*cvother));

invref = inv(diag(cvref));
invother = inv(diag(cvother));

for k = 1:numcofi
[k numcofi]

closest = findnearest(cofi(k,:),othc,cvref);

if i < j
orient=1;
elseif j > i
orient=2;
end%ifi


[N,p] = computeNP(cofi(k,:),cmref,cvref,closest,cmother, ..
cvother,detref,detother,invref,invother,.5,.5,orient);

testN = sum(N);

% One should only get imaginary numbers if an incorrectly classified
% sample slipped past the guards:
if imag(N)~=0
'Uh-oh, N is complex'
numberofimag = numberofimag+1;
else
EDBFM = EDBFM + (1/numcofi)*(N*N');
end
end%fork

eval(['E' num2str(i) num2str(j) ' = EDBFM;'])

end%ifj~=i
end%forj
end%fori

stat = 'calculate EDBFM '

%  Calculate EDBFM

EDBFM = zeros(n,n);

% Assume aprioris are equal
p1 =1/classes;
p2 =1/classes;

for i = 1:4
for j=1:4
if j~=i
eval(['EDBFM = EDBFM + p1*p2*E' num2str(i) num2str(j) ';'])
end%ifj~=i
end%forj
end%fori

[V,d] = eig(EDBFM);


rankdbfm = rank(EDBFM)

save results256full EDBFM rankdbfm d V numberofimag
```

C.2.2 Reclassification in Transformed Space.

```
% This program performs extraction and testing
% for a four class problem in the transformed feature space
clear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load in training and testing data sets
train = 1;
transform = 1;
load trnfa256
load trnf2256
load trn9a256
load trnaa256

load tst9a256
load tstaa256
load tstf2256
load tstfa256

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load in EDBFM information

load results256full

results = zeros(32,4);
aa = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% kk is the looping parameter.  For the results in the thesis,
% I generarated them using the following numbers of features:
% 1, 3 , 5, 7, 9,.. 19, 21, 50, 100, 200,  250, 256

for kk = [1:2:21 50:50:250 256];
correct1 = 0;
correct2 = 0;
correct3 = 0;
correct4 = 0;

aa
if train == 1

trnc1 = trn9a256(:,:);
trnc2 = trnaa256(:,:);
trnc3 = trnf2256(:,:);
trnc4 = trnfa256(:,:);

% TRN set has been previously aligned

% Select how many relevant eigenvectors
% you wish to use

relnum = kk;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Recall that V holds the eigenvectors of the EDBFM, ordered
% by magnitude of its eigenvalue.  This step selects the
% the number to use.

Vt = V(:,256-relnum+1:256);

% During troubleshooting, one may elect not to transform
% at all
if transform == 0
Vt = diag(ones(1,256));
end

% Perform transformation, based on the EDBFM
'transforming training data'
trnc1 = trnc1 * Vt;
trnc2 = trnc2 * Vt;
trnc3 = trnc3 * Vt;
trnc4 = trnc4 * Vt;

% Implement the training portion of a Gaussian classifier,
% recursively computing computing mean and variances.
% Note that the data sets were previously aligned before
% extraction from the AGC algorithm

'training 9a'
mx9a = trnc1(1,:);
vx9a = zeros(1,size(trnc1,2));
for i = 2:size(trnc1,1)
currx = trnc1(i,:);

mx9a = ((i-1)*mx9a+currx)/i;

vx9a = (((i-1)*vx9a)/i)+((mx9a-currx).^2)/(i-1);
end
```

```
'training aa'
mxaa = trnc2(1,:);
vxaa = zeros(1,size(trnc2,2));
for i = 2:size(trnc2,1)
currx = trnc2(i,:);

mxaa = ((i-1)*mxaa+currx)/i;

vxaa = (((i-1)*vxaa)/i)+((mxaa-currx).^2)/(i-1);
end


'training f2'
mxf2 = trnc3(1,:);
vxf2 = zeros(1,size(trnc3,2));
for i = 2:size(trnc3,1)
currx = trnc3(i,:);

mxf2 = ((i-1)*mxf2+currx)/i;

vxf2 = (((i-1)*vxf2)/i)+((mxf2-currx).^2)/(i-1);
end

'training fa'
mxfa = trnc4(1,:);
vxfa = zeros(1,size(trnc4,2));
for i = 2:size(trnc4,1)
currx = trnc4(i,:);

mxfa = ((i-1)*mxfa+currx)/i;

vxfa = (((i-1)*vxfa)/i)+((mxfa-currx).^2)/(i-1);
end

% Calculate bias terms:
b9a = sum(log(vx9a));
baa = sum(log(vxaa));
bf2 = sum(log(vxf2));
bfa = sum(log(vxfa));

end%iftrain


c1test = tst9a256;
c2test = tstaa256;
c3test = tstf2256;
c4test = tstfa256;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Transform test data, as above

% Perform transformation, based on the EDBFM
% 'transforming test data'

c1test = c1test * Vt;
c2test = c2test * Vt;
c3test = c3test * Vt;
c4test = c4test * Vt;

% In all of the following ''dxy'' implies one is comparing
% the xth class mean with an exemplar from class y.  Note that
% 1-4 corresponds to 9a-fa as shown at the beginning of the EDBFM
% analysis program

% 'testing 9a'

for i = 1:size(c1test,1)
[i size(c1test,1)];

xt = c1test(i,:);

% Give this subroutine the current exemplar
% the mean for the respective class, the
% variance for the respective class, the
% bias for the respective class, and the
% number of features used.

d11 = decisionwshiftkk(xt,mx9a,vx9a,b9a,kk);
d21 = decisionwshiftkk(xt,mxaa,vxaa,baa,kk);
d31 = decisionwshiftkk(xt,mxf2,vxf2,bf2,kk);
d41 = decisionwshiftkk(xt,mxfa,vxfa,bfa,kk);

% Now classify the data, saving relevant information for later

[dum,idx] = min([d11 d21 d31 d41]);
if idx == 1
```

```
correct1 = correct1 + 1;
end
correct1;
end

% 'testing aa'

for i = 1:size(c2test,1)
[i size(c2test,1)];

xt = c2test(i,:);

d11 = decisionwshiftkk(xt,mx9a,vx9a,b9a,kk);
d21 = decisionwshiftkk(xt,mxaa,vxaa,baa,kk);
d31 = decisionwshiftkk(xt,mxf2,vxf2,bf2,kk);
d41 = decisionwshiftkk(xt,mxfa,vxfa,bfa,kk);

%  Now classify the data, saving relevant information for later

[dum,idx] = min([d11 d21 d31 d41]);
if idx == 2
correct2 = correct2 + 1;
end
correct2;
end

% 'testing f2'
for i = 1:size(c3test,1)
[i size(c3test,1)];

xt = c3test(i,:);

d11 = decisionwshiftkk(xt,mx9a,vx9a,b9a,kk);
d21 = decisionwshiftkk(xt,mxaa,vxaa,baa,kk);
d31 = decisionwshiftkk(xt,mxf2,vxf2,bf2,kk);
d41 = decisionwshiftkk(xt,mxfa,vxfa,bfa,kk);

%  Now classify the data, saving relevant information for later

[dum,idx] = min([d11 d21 d31 d41]);
if idx == 3
correct3 = correct3 + 1;
end
correct3;
end


% 'testing fa'

for i = 1:size(c4test,1)
[i size(c4test,1)];

xt = c4test(i,:);

d11 = decisionwshiftkk(xt,mx9a,vx9a,b9a,kk);
d21 = decisionwshiftkk(xt,mxaa,vxaa,baa,kk);
d31 = decisionwshiftkk(xt,mxf2,vxf2,bf2,kk);
d41 = decisionwshiftkk(xt,mxfa,vxfa,bfa,kk);

%  Now classify the data, saving relevant information for later

[dum,idx] = min([d11 d21 d31 d41]);
if idx == 4
correct4 = correct4 + 1;
end
correct4;
end


results(aa,1) = correct1;
results(aa,2) = correct2;
results(aa,3) = correct3;
results(aa,4) = correct4;


aa = aa+1;

if aa == 1
results(1,:)
end

end%forkk

save tran4rateswshifts results
```

C.2.3 Reclassification in Original Space.

```
% This program performs extraction and testing
% for a four class problem in the original feature space
clear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load in training and testing data sets
train = 1;
transform = 1;
load trnfa256
load trnf2256
load trn9a256
load trnaa256

load tst9a256
load tstaa256
load tstf2256
load tstfa256

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load in EDBFM information

load results256full

results = zeros(32,4);
aa = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% kk is the looping parameter. For the results in the thesis,
% I generarated using the following numbers of features:
% 1, 3 , 5, 7, 9,.. 19, 21, 50, 100, 200, 250, 256

for kk = [1:2:21 50:50:250 256];
correct1 = 0;
correct2 = 0;
correct3 = 0;
correct4 = 0;

aa

% Evaluate V to choose significant features
% Note, use an average of the ten most significant
% eigenvectors

useus = abs(V(:,247:256)');

useus = mean(useus);

featselect = [];
for numfeats = 1:kk

[wch,idx] = max(useus);
useus(idx) = -100;

featselect = [featselect idx];

end

if train == 1

trnc1 = trn9a256(:,featselect);
trnc2 = trnaa256(:,featselect);
trnc3 = trnf2256(:,featselect);
trnc4 = trnfa256(:,featselect);

% TRN set has been previously aligned

% Select how many relevant eigenvectors
% you wish to use

relnum = kk;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Implement the training portion of a Gaussian classifier,
% recursively computing computing mean and variances.
% Note that the data sets were previously aligned before
% extraction from the AGC algorithm

'training 9a'
mx9a = trnc1(1,:);
vx9a = zeros(1,size(trnc1,2));
for i = 2:size(trnc1,1)
currx = trnc1(i,:);

mx9a = ((i-1)*mx9a+currx)/i;

vx9a = (((i-1)*vx9a)/i)+((mx9a-currx).^2)/(i-1);
end
```

```
'training aa'
mxaa = trnc2(1,:);
vxaa = zeros(1,size(trnc2,2));
for i = 2:size(trnc2,1)
currx = trnc2(i,:);

mxaa = ((i-1)*mxaa+currx)/i;

vxaa = (((i-1)*vxaa)/i)+((mxaa-currx).^2)/(i-1);
end


'training f2'
mxf2 = trnc3(1,:);
vxf2 = zeros(1,size(trnc3,2));
for i = 2:size(trnc3,1)
currx = trnc3(i,:);

mxf2 = ((i-1)*mxf2+currx)/i;

vxf2 = (((i-1)*vxf2)/i)+((mxf2-currx).^2)/(i-1);
end

'training fa'
mxfa = trnc4(1,:);
vxfa = zeros(1,size(trnc4,2));
for i = 2:size(trnc4,1)
currx = trnc4(i,:);

mxfa = ((i-1)*mxfa+currx)/i;

vxfa = (((i-1)*vxfa)/i)+((mxfa-currx).^2)/(i-1);
end

% Calculate bias terms:
b9a = sum(log(vx9a));
baa = sum(log(vxaa));
bf2 = sum(log(vxf2));
bfa = sum(log(vxfa));

end%iftrain


c1test = tst9a256;
c2test = tstaa256;
c3test = tstf2256;
c4test = tstfa256;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Perform feature selection on the test sets

c1test = tst9a256(:,featselect);
c2test = tstaa256(:,featselect);
c3test = tstf2256(:,featselect);
c4test = tstfa256(:,featselect);

% In all of the following ''dxy'' implies one is comparing
% the xth class mean with an exemplar from class y.  Note that
% 1-4 corresponds to 9a-fa as shown at the beginning of the EDBFM
% analysis program

% 'testing 9a'

for i = 1:size(c1test,1)
[i size(c1test,1)];

xt = c1test(i,:);

% Give this subroutine the current exemplar
% the mean for the respective class, the
% variance for the respective class, the
% bias for the respective class, and the
% number of features used.

d11 = decisionwshiftkk(xt,mx9a,vx9a,b9a,kk);
d21 = decisionwshiftkk(xt,mxaa,vxaa,baa,kk);
d31 = decisionwshiftkk(xt,mxf2,vxf2,bf2,kk);
d41 = decisionwshiftkk(xt,mxfa,vxfa,bfa,kk);

% Now classify the data, saving relevant information for later

[dum,idx] = min([d11 d21 d31 d41]);
if idx == 1
correct1 = correct1 + 1;
end
correct1;
```

```
end

% 'testing aa'

for i = 1:size(c2test,1)
[i size(c2test,1)];

xt = c2test(i,:);

d11 = decisionwshiftkk(xt,mx9a,vx9a,b9a,kk);
d21 = decisionwshiftkk(xt,mxaa,vxaa,baa,kk);
d31 = decisionwshiftkk(xt,mxf2,vxf2,bf2,kk);
d41 = decisionwshiftkk(xt,mxfa,vxfa,bfa,kk);

%  Now classify the data, saving relevant information for later

[dum,idx] = min([d11 d21 d31 d41]);
if idx == 2
correct2 = correct2 + 1;
end
correct2;
end

% 'testing f2'

for i = 1:size(c3test,1)
[i size(c3test,1)];

xt = c3test(i,:);

d11 = decisionwshiftkk(xt,mx9a,vx9a,b9a,kk);
d21 = decisionwshiftkk(xt,mxaa,vxaa,baa,kk);
d31 = decisionwshiftkk(xt,mxf2,vxf2,bf2,kk);
d41 = decisionwshiftkk(xt,mxfa,vxfa,bfa,kk);

%  Now classify the data, saving relevant information for later

[dum,idx] = min([d11 d21 d31 d41]);
if idx == 3
correct3 = correct3 + 1;
end
correct3;
end

% 'testing fa'

for i = 1:size(c4test,1)
[i size(c4test,1)];

xt = c4test(i,:);

d11 = decisionwshiftkk(xt,mx9a,vx9a,b9a,kk);
d21 = decisionwshiftkk(xt,mxaa,vxaa,baa,kk);
d31 = decisionwshiftkk(xt,mxf2,vxf2,bf2,kk);
d41 = decisionwshiftkk(xt,mxfa,vxfa,bfa,kk);

%  Now classify the data, saving relevant information for later

[dum,idx] = min([d11 d21 d31 d41]);
if idx == 4
correct4 = correct4 + 1;
end
correct4;
end

results(aa,1) = correct1;
results(aa,2) = correct2;
results(aa,3) = correct3;
results(aa,4) = correct4;

aa = aa+1;

if aa == 1
results(1,:)
end

end%forkk

save orig4rateswshifts results
```

C.2.4 Subroutines.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CHITEST SUBROUTINE
%  This subroutine performs the intraclass chitest
%  The "col" variable corresponds to the appropriate
%  class being compared against
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[goodonly] = chitest(x,chialpha,col)
%x = d9a;
%chialpha = chialpha;
%col =1 ;

goodonly = [];
%size(x)
for i = 1:size(x,1)

if x(i,col) < chialpha

goodonly = [goodonly; i];
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LTEST SUBROUTINE
%  Same as above, but Lmin insures at least a minimum
%  number of exemplars are used from the other classes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[goodonly] = Ltest(x,chialpha,col,Lm1)

goodonly = [];
wx =size(x,1);
for i = 1:size(x,1)
[i,ans];
if x(i,col) < chialpha
goodonly = [goodonly; i];
end
end

chk = size(goodonly,1);

if chk < Lm1
[dum wch] = sort(x);
goodonly = wch(1:Lm1);
end%if

size(wch);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FINDNEAREST SUBROUTINE
% This subroutine finds the nearest in the sense of Mahalanobis distance,
%  as prescribed by Lee and Landgrebe
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[winner] = findnearest(reference,testsigs,var)

[numsigs,feats] = size(testsigs);
bestmatch = 0;
dists = zeros(numsigs,1);
var=var;


for i = 1:numsigs
i;
xtrial = testsigs(i,:);
[lx,wx] = size(xtrial);
[lref,wref] = size(reference);

dists(i) = (var.^(-1)).*(xtrial-reference)*(xtrial-reference)';

end

[Y,maxind] = min(dists);

winner = testsigs(maxind,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% COMPUTENP SUBROUTINE
%  This subroutine implements Lee's equations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  The difference between this and the test function
% is that one assumes uncorrelated features (variances
%  are in a vector)

x1;x2;m1;m2;v1;v2;
m1 = m1';m2 = m2';
v1 = v1';v2 = v2';
x1 = x1';x2 = x2';
```

```
v0 = x1;
v = x2-x1;

sm1=size(m1);
sinv1=size(inv1);
sm2=size(m2);
sinv2=size(inv2);

c = .5 * (m1'*inv1*m1 - m2'*inv2*m2) + .5*log( det1/det2 );
cprime = .5*v0' * (inv1-inv2) * v0 - (m1'*inv1-m2'*inv2)*v0 + c;

b = v0'*(inv1-inv2)*v - (m1'*inv1-m2'*inv2)*v;
a = .5 * v' * (inv1-inv2) * v;

t = log(prob1/prob2);

% Note: this step finds the appropriate root when one
% as two to choose from
if a == 0
u = (t-cprime)/b;
else
u1 = (-b + sqrt(b^2-4*a*(cprime-t)))/(2*a);
u2 = (-b - sqrt(b^2-4*a*(cprime-t)))/(2*a);
if orient==1
u = min([u1 u2]);
else
u = max([u1 u2]);
end
end

p = u*v+v0;

N = (inv1-inv2)*p + (inv2*m1 - inv1*m2);
P=p;

N = N/sqrt(N'*N);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DECISIONWSHIFTSKK SUBROUTINE
% This subroutine implements the classification
% scheme like that of the AGC. In other words,
% exemplars are shifted up to 18 bins in either
% direction and matched to each template and then
% a winner is declared in the main program.
% This subroutine just does the correlation/shifting
% portion. kk represents how many features are
% being used. In this particular version of the
% subroutine a single number is returned, but the program
% may be easily modified to return the shifted signature
% if necessary
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[dist] = decisionwshift(x,template,var,b,kk)
% This routine performs a circular convolution
% on a set of signals vs a template

lbound = -18;
ubound = 18;

if abs(lbound) > kk
lbound = -kk+2;
ubound = kk-2;
end

if kk == 2
lbound = 0;
ubound = 0;
end

if kk == 1
lbound = 0;
ubound = 0;
end

[wx,lx] = size(x);
xtrial = zeros(1,lx);

alignedsig = zeros(1,lx);
shiftstore = zeros(wx,1);

reference = template;
%reference = reference/sqrt(reference*reference');
%reference(1:5)


% Find offsets via circular shifting

xtemp = x;
```

```
% xtemp = xtemp/sqrt(xtemp*xtemp');
%xtemp(1:5)

mdist = zeros(1,tryshift-lbound+1);
bestshift = 0;

for tryshift = lbound:ubound
aa = tryshift-lbound+1;

if tryshift < 0
xtrial(aa,:) = [xtemp(-tryshift+1:lx) xtemp(1:-tryshift)];
elseif tryshift == 0
xtrial(aa,:) = xtemp;
elseif tryshift > 0
xtrial(aa,:) = [xtemp(lx-tryshift+1:lx) xtemp(1:lx-tryshift)];
end


mdist(aa) =  ..
.5*(var.^(-1)).*(xtrial(aa,:)-reference)*(xtrial(aa,:)-reference)'+.5 * b;
end%for

[dist,bestshift] = min(mdist);

alignedsig = xtrial(bestshift,:);

bestshift = bestshift-1;

shiftstore = bestshift;
```

Bibliography

1. *ARTI Phase III Data.* Technical Report, WL/AARM: Veda Incorporated, October 1992.
2. Allen, Arnold O. *Probability, Statistics, and Queuing Theory With Computer Science Applications.* Academic Press, Inc., 1990.
3. Baras, John and Sheldon Wolk. "Model Based Automatic Target Recognition from High Range Resolution Radar Returns," *SPIE, 2234*:57–66 (1994).
4. Benveniste, A. and others. "Multiscale Sytem-Theory," *IEEE Transactions on Circuits and Systems, 41*(1):2–15 (January 1994).
5. Chandrasekaran, B. *IEEE Transaction on Information Theory, IT-17*:452 (July 1971).
6. Chang, Tianhorng and C.-C. Jay Kuo. "Texture Analysis and Classification with Tree-Structured Wavelet Transform," *IEEE Transactions on Image Processing, 2*(4):429–441 (October 1993).
7. Chou, K. and others. "Multiscale Recursive Estimation, Data Fusion, and Regularization," *IEEE Transactions on Automatic Control, 39*(3):464–478 (March 1994).
8. Cohen, Albert and Ingrid Daubechies. "Wavelets on the Interval and Fast Wavelet Transforms," *Applied and Computational Harmonic Analysis* (1993).
9. Coifman, Ronald and Naoki Saito. "Constructions of Local Orthonormal Bases for Classification and Regression," *Comptes Rendus De L Academie Des Sciences Serie I-Mathematique, 319*(2):191–196 (July 1994).
10. Coifman, Ronald R. and Mladen Victor Wickerhauser. "Entropy-Based Algorithms for Best Basis Selection," *IEEE Transactions on Information Theory, 38*(2):713–718 (March 1992).
11. Cover, T. M. "Geometrical and Statistical Properties of Linear Inequalities With Applications in Pattern Recognition," *IEEE Transactions in Electronic Computation, EC-14*:326–334 (June 1965).
12. DelMarco, S. and J. Weiss. "M-Band Wavepacket-Based Transient Signal detector Using a Translation-Invariant Wavelet Transform," *Optical Engineering, 33*(7):2175–2182 (July 1994).
13. Devijver, P. and J. Kittler. *Pattern Recognition, A Statistical Approach.* Prentice Hall, 1982.
14. DeWall, Rob. "Issues on the Implementation of the AGC." Conversations Aug-Oct 1994.
15. Dewitt, Mark R. *High Range Resolution Radar Target Identification Using the Prony Model and Hidden Markov Models.* MS thesis, Air Force Institute of Technology, 1992.
16. Duda, R. and P. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

17. Estes, S. E. *Measurement Selection for Linear Discriminants Used in Pattern Classification.* Technical Report RJ-331, San Jose Research Lab, April 1965.

18. Foley, Donald H. "Considerations of Sample and Feature Size," *IEEE Transactions on Information Theory, IT-18*(5):618–626 (September 1972).

19. Fukunaga, Keinosuke. *Introduction to Statistical Pattern Recognition.* Academic Press, Inc., 1990.

20. Fukunaga, Keinosuke and Donald M. Hummels. "Bayes Error Estimation Using Parzen and *k*-NN Procedures," *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-9*(5):634–643 (September 1987).

21. Geman, Stuart and others. "Neural Networks and the Bias/Variance Dilemma," *Neural Computation, 4*(1):1–58 (1992).

22. Gersho, A. and R. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Press, 1991.

23. Jain, Anil and Jianchang Mao. "Neural Networks and Pattern Recognition." *Computational Intelligence Imitating Life* edited by Jacek M. Zurada and others, IEEE Press, 1994.

24. Kanal, L. and B. Chandrasekaran. "On Dimensionality and Sample Size in Statistical Pattern Classification." *Proceedings: 1968 National Electronics Conference.* 2–7. 1968.

25. Kocur, Katherine. *Computer-Aided Breast Cancer Diagnosis.* MS thesis, Air Force Institute of Technology, 1994.

26. Kouba, Eric T. *Recurrent Neural Networks for Radar Target Identification.* MS thesis, Air Force Institute of Technology, December 1992.

27. Lachenbruch, P. A. and R. Mickey. *Technometrics, 10.* 1968.

28. Lee, Chulhee and David A. Landgrebe. "Decision Boundary Feature Extraction for Nonparametric Classification," *IEEE Transactions on Systems, Man, and Cybernetics, 23*(2):433–444 (March 1993).

29. Lee, Chulhee and David A. Landgrebe. "Feature Extraction Based on Decision Boundaries," *IEEE Transactions on Pattern Analysis and Machine Intelligence, 15*(4):388–400 (April 1993).

30. Lippmann, Richard P. "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine, 4* (1987).

31. Mallat, Stephane G. "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence, 11*(7):674–692 (July 1989).

32. Martin, Curtis E. *Non-Parametric Bayes Error Estimation for UHRR Target Identification.* MS thesis, Air Force Institute of Technology, 1993.

33. Martin, Curtis E. and others. "Neural Network Bayes Error Estimation." *IEEE World Congress on Computational Intelligence.* June 1994.

34. Mitchell, Richard and Rob DeWall. *Overview of High Range Resolution Radar Target Identification.* Technical Report, Wright Laboratories, 1994.

35. Myers, Lemuel. *Image Interpretation and Enhancement for the Visually Impaired.* MS thesis, Air Force Institute of Technology, 1994.

36. Oxley, Mark. "Interpretation of the outer product." Conversation, 9 Nov 94.

37. Papoulis, Anthonolis. *Probability, Stochastics, and Random Processes* (3 Edition). McGraw Hill, 1991.

38. Parsons, Thomas. *Voice and Speech Processing.* McGraw Hill, Inc., 1987.

39. Rogers, Steven K. and others. *An Introduction to Biological and Artificial Neural Networks.* Wright Patterson AFB, OH: Air Force Institute of Technology, 1990.

40. Ruck, Dennis W. and others. "The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function," *IEEE Transactions on Neural Networks* (1990).

41. Schalkoff, Robert J. *Pattern Recognition: Statistical, Structural, and Neural Approaches.* John Wiley & Sons, Inc., 1992.

42. Steppe, Jean and others. "Feature and Model Selection in Feedforward Neural Networks," *Submitted to IEEE Transactions on Neural Networks* (1994).

43. Steppe, Jean M. *Feature and Model Selection in Feedforward Neural Networks.* PhD dissertation, Air Force Institute of Technology, 1994.

44. Stimson, George W. *Introduction to Airborne Radar.* Hughes Aircraft Corporation, 1983.

45. Strang, Gilbert. *Linear Algerbra and its Applications* (3rd Edition). Harcourt Brace Jovanovich College Publishers, 1988.

46. Suzuki, Laura. "A Simplistic Speaker ID Method." In Partial Fulfillment of the AFIT/ENG PhD Minor Examination Requirement, June 1993.

47. Vapnik, V. *Estimation of Dependance Based on Empirical Data.* Springer Verlag, 1982.

Vita

Captain Christopher Lawrence Eisenbies was born on 9 July 1966 in Raleigh, North Carolina. He graduated *cum laude* from Duke University in 1989 with a Bachelor of Science in Electrical Engineering, and accepted a commission as a Second Lieutenant in the United States Air Force. In August of 1989, Lieutenant Eisenbies was assigned to Phillips Laboratory, Kirtland Air Force Base, New Mexico, where he tested Air Force systems with respect to electromagnetic pulse and high power microwaves. In May of 1993, Lieutenant Eisenbies was assigned to the Air Force Institute of Technology, Wright Patterson Air Force Base, Ohio, to pursue a Master of Science degree in Electrical Engineering. Upon completion of that assignment, Captain Eisenbies will proceed to the National Aerospace Intelligence Center in the office of Technical Assessment.

Permanent address: 79 Larchmere Drive
Beavercreek, OH 45440

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1994	Master's Thesis

4. TITLE AND SUBTITLE
Classification of Ultra High Range Resolution Radar Using Decision Boundary Analysis

5. FUNDING NUMBERS

6. AUTHOR(S)
Christopher Lawrence Eisenbies

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/94D-07

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Capt. Thomas Burns
USAF Wright Laboratory
WL/AARA
Wright Patterson AFB, OH 45433-7001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This thesis examines the discrimination of targets with Ultra High Range Resolution (UHRR) radar data. Using these measured signals from frontal aspect angles of four aircraft classes, the baseline performance of the Adaptive Gaussian Classifier (AGC) is tested with respect to aligning exemplars to templates. Alignment plays a crucial role in the AGC's classification performance which can degrade by 11% for a target class. The AGC is compared to non-parametric classifiers, but no statistically significant degradation of performance is found. Data separability is analyzed by bounding the Bayes error. The data is well separated in a statistical sense. A feature selection algorithm, based on analysis of the decision boundary, is applied to find a reduced feature set, which are linear combinations of the original features. These features are optimized with respect to classification error rather than reconstruction error. This technique is extended to deduce the relevant features in the *original* feature space. Fewer than 5% of the features in the original feature space may be used to attain an improved classification rate. This new method is a true reduction of features and shows improvement up to 15%. Discrimination of UHRR radar signatures using a multiresolution analysis is proposed. The decision boundary analysis chooses relevant wavelet scales with respect to classification. Some improved performance against an entropy based measure is observed for limited feature sets. The technique developed here successfully chooses the scale that causes classification performance to peak within 5% of the performance in the full-dimensional or reduced-dimensional UHRR radar signature space.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Gaussian classifier, Decision boundary analysis, Feature selection, Target recognition	120
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.